Design • Build • Program • Learn • Compete

# SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

January 2012

# MEET MADEUSA

◆ Is Your Robot "Jamming?"

Experiment with this novel technique to grip irregular shaped objects

◆ The Perfect Solderless Breadboard

A look at Digilent's Electronics Explorer

◆ TurtleBot

ROS meets Kinect meets Create

◆ The History Of Robot Combat

Humble beginnings to multinational sensation

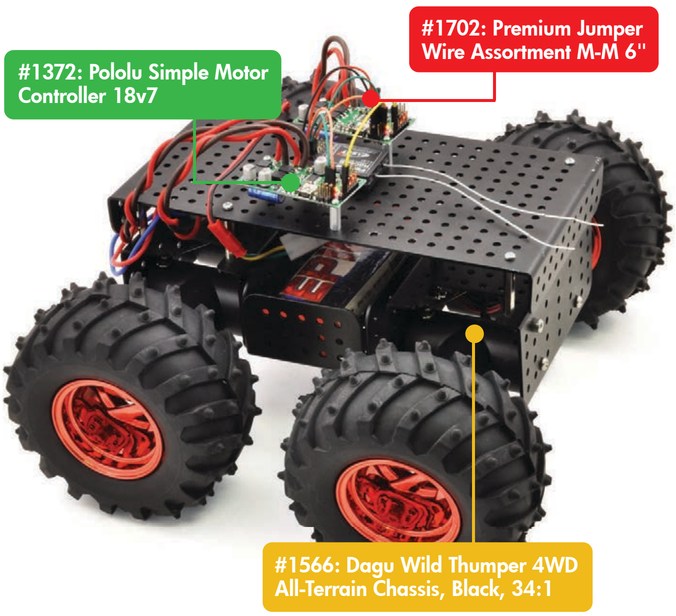
U.S. $5.50 CANADA $7.00

01>

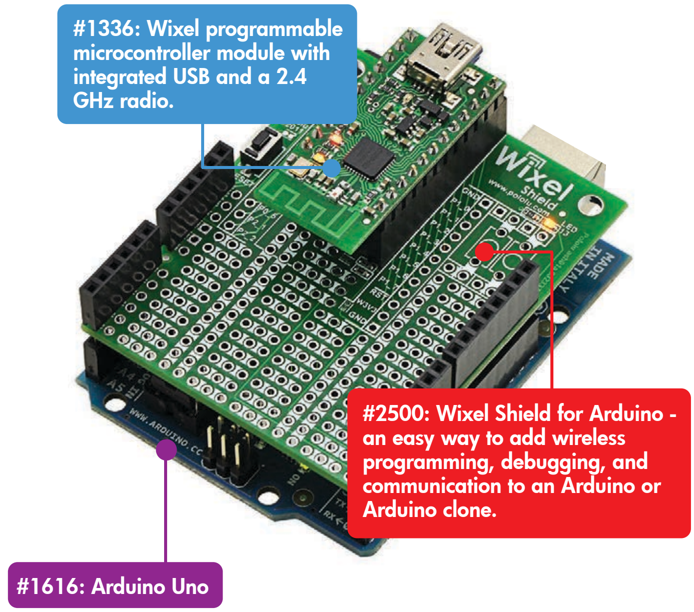
0 71486 02422 4


LISTEN UP!
YOUR BOT IS TALKING!
Knowing what your robot is doing can make a world of difference when you are competing. Our HTS-SS Blue Sensor station and series of sensors puts valuable GPS, speed, altitude, RPM, temperature, voltage and current information at your fingertips. Couple our HTS-Voice with your Hitec radio and your bot will actually tell you the data you want to hear. Plug the HTS-Navi into your PC and analyze your data at the end of the day!
HITEC 2.4GHz
ADAPTIVE FREQUENCY HOPPING SPREAD SPECTRUM
GET EQUIPPED!
GET EMPOWERED!
GET HITEC!
HTS-VOICE
HTS-SS BLUE
HTS-NAVI
[Current Sensor]
Current : 95.2A / C200
Wattage : 1332.8W
Warning (Hi) : 100A
3/3
MAX 98.0A
MAX 1421W
RST
AURORA 9
AURORA 9
Installed with the
HTS-VOICE
HITEC
12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

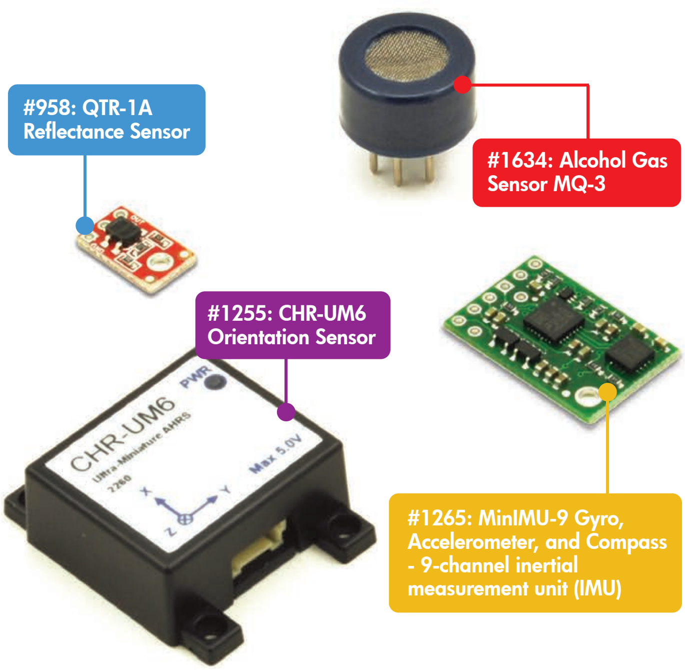
DIY Projects:
## Wild Thumper-Based Robot



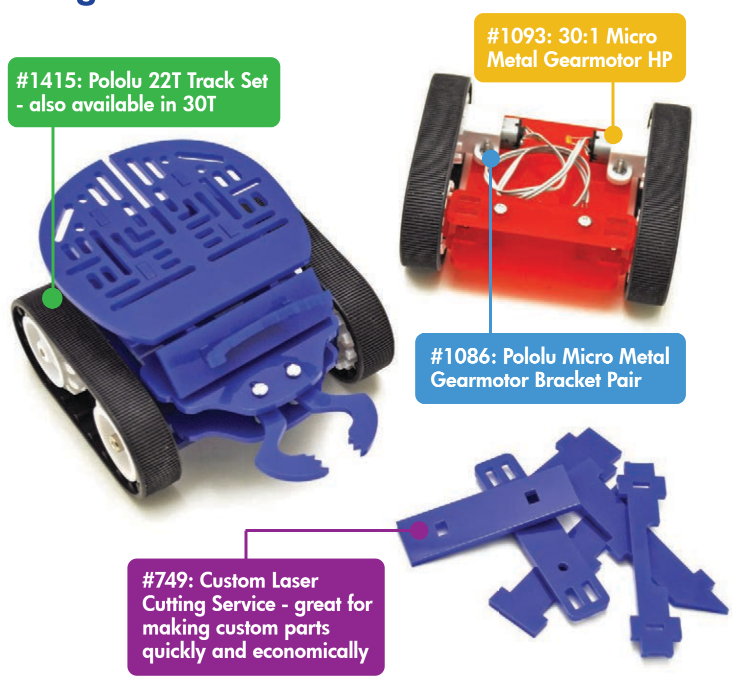

Programmable Controllers:
## Wixel and Wixel Shield




Sensors:
## Orientation, Reflectance and More




Custom Laser Cutting:
## Design Your Own Chassis




**Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products - from actuators to wireless modules - that can help you take your robot from idea to reality.**

Pololu
Engage Your Brain

Find these products and more at **www.pololu.com**

# SERVO MAGAZINE

01.2012
VOL. 10 NO. 1

PAGE 76

# Columns



# The Combat Zone...

## Features

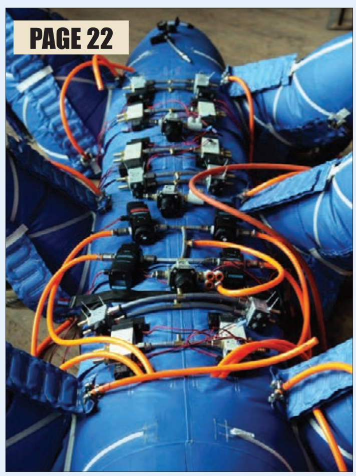


## Events



# Departments

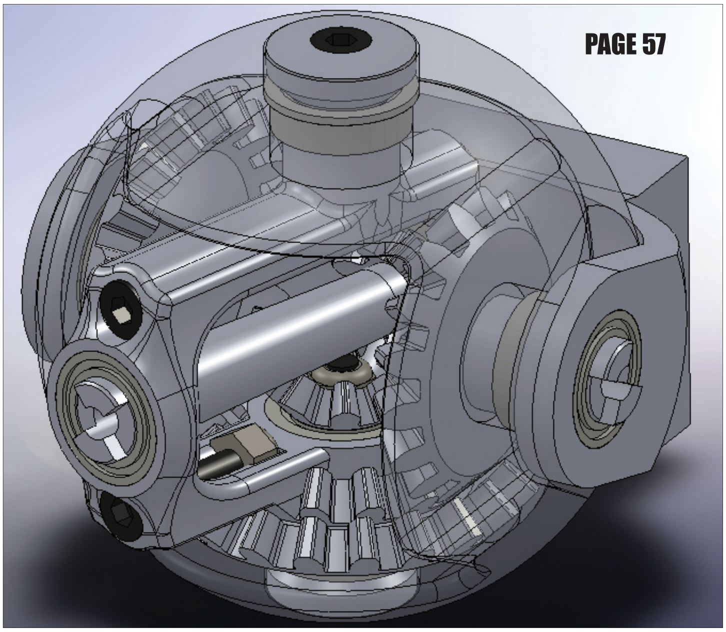




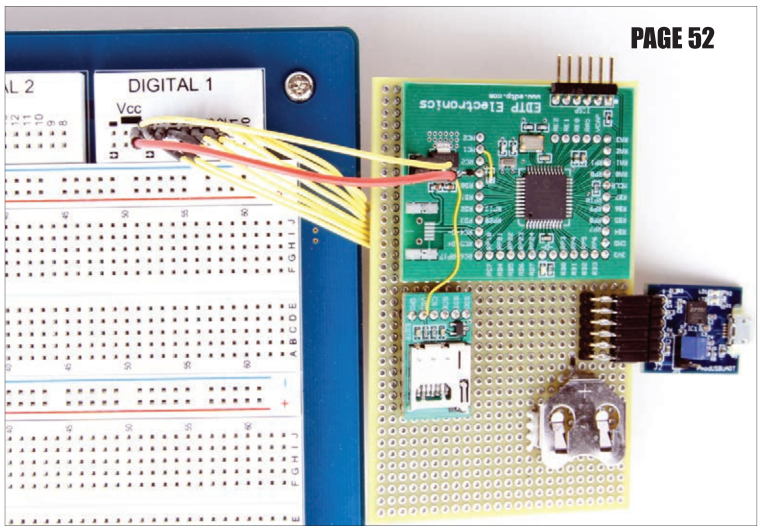
PAGE 22

*SERVO Magazine* **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

# In This Issue ...

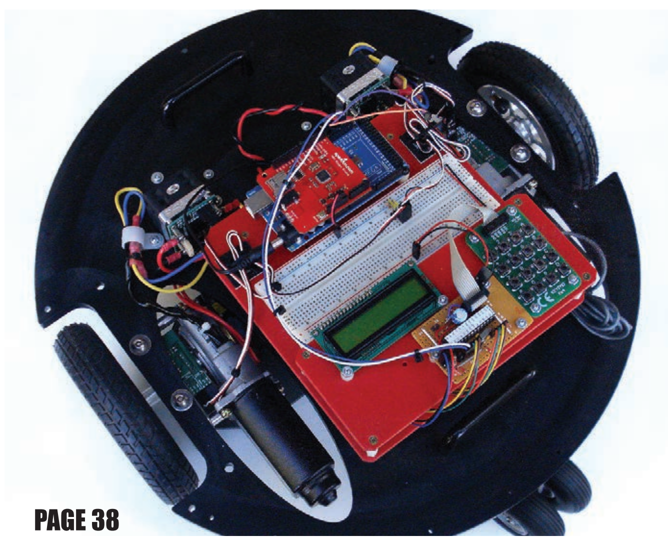

PAGE 57


PAGE 44



PAGE 52




PAGE 38

# Mind / Iron

by Bryan Bergeron, Editor

## Robot School — Will Neurosynaptic Chips Put an End to Traditional Programming?

IBM's new cognitive computing chip — which is loosely based on the structure and function of a biological nervous system — may signal the eventual end of programming as we know it. The chips — though built of silicon — are designed to learn from experience, much like the way neural networks correlate outcomes with input variables. The chip is about to be put to the test by researchers from Columbia University, Cornell University, the University of California, and the University of Wisconsin.

So, assuming IBM and the team of academicians are successful at creating practical solutions with these chips, what does it portend for robotics? For one, it means that the days of implicit calls to sensors and interrupts are a thing of the past. Perhaps programming your vacuum cleaner in five years will consist of leading your vacuum around with a choker chain — similar to that used with a dog — while you point out areas of your home that need special attention. Positive feedback might be in the form of verbal acknowledgements, and a quick yank of the choker chain could signal your dissatisfaction.

It could also mean that your automobile's autopilot automatically and continually models your driving so as to emulate your every move in every conceivable situation. That could be a bad thing, depending on how you drive. Perhaps there will be a market for driver programs modeled after well-known racecar drivers, or perhaps neurosynaptic chips trained by insurance companies. Get your 10% discount on insurance if you agree to drive the Allstate way, for example.

Does this mean you should give up on learning C++ and sit back and wait for the eager robot students to arrive? I don't think so. The early systems — which won't be out for a few years — will undoubtedly be hybrid systems comprised of neurosynaptic chips interfaced to traditional microcontrollers and microprocessors that follow Harvard or von Neumann architectures. Plus, these chips will be programmed the old fashioned way with C+, Basic, or some other language.

Clearly, the new chip could signal the beginning of an era of robotics 'for the rest of us.' Virtually no learning curve and no need for computer literacy will certainly lower the adoption barrier. The only issues remaining are cost, form factor, and legislation. The cost of a robotic home nurse might initially be affordable by only a select few, but with time, they'll be as affordable as flat screen TVs. Form factor — meaning do you want a robotic dog, person, or cooking machine — will depend on where you are in life and your health.

Legislation, of course, will dictate the limits of the new technology. A robotic surgeon will likely be held up by the AMA and FDA for 15 or 20 years while it's evaluated for medical efficacy. Scantily clad, humanoid robots will likely be illegal in some settings. I'm not sure I'd trust a robotic cabin crew on a cross-country flight, but I suppose that's inevitable as well.

Assuming IBM's chips live up to expectations, what do you see in store for robotics in the next decade? SV

FOR THE ROBOT INNOVATOR **SERVO** MAGAZINE

Published Monthly By
**T & L Publications, Inc.**
430 Princeland Ct., Corona, CA 92879-1300
**(951) 371-8497**
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
**www.servomagazine.com**

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

**PUBLISHER**
Larry Lemieux
**publisher@servomagazine.com**

**ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING**
Robin Lemieux
**display@servomagazine.com**

**EDITOR**
Bryan Bergeron
**techedit-servo@yahoo.com**

**CONTRIBUTING EDITORS**
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Kevin Berry, Gordon McComb, Fred Eady, Greg Intermaggio, Bryce Woolley, Evan Woolley, Charles Ford, Matt Bunting, Mike Jeffries, Andrea Suarez, Dave Graham, Morgan Berry

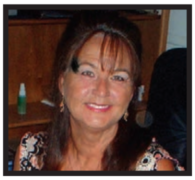
**CIRCULATION DIRECTOR**
Tracy Kerley
**subscribe@servomagazine.com**

**MARKETING COORDINATOR WEBSTORE**
Brian Kirkpatrick
**sales@servomagazine.com**

**WEB CONTENT**
Michael Kaudze
**website@servomagazine.com**

**ADMINISTRATIVE ASSISTANT**
Debbie Stauffacher

**PRODUCTION/GRAPHICS**
Shannon Christensen



# Robytes

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com.

by Jeff and Jenn Eckert

*Teotronica's jazz robot performs at the piano.*

## Meanwhile, Across the Pond

Before you imagine that a performing robot skeleton is a totally unique concept, consider the Teotronica Jazz Robot produced by the Italian electronic design and repair company Teotronica (**www.teotronica.it**). He doesn't appear to be available in the US yet, but this guy "has no difficulty in running even the most demanding music and any kind of music (blues, jazz, rock, classical, etc.)." The entertainment bots are available for pretty much any kind of public or private event, and interested parties are directed to request a free estimate for rent or sale at info@teotronica.it.

## The Scoop on Geoff


*Geoff Peterson — TV's strangest sidekick.*

If you have a tendency toward insomnia and can filter through a thick Scottish accent, you may be familiar with Geoff Peterson, the robotic skeleton sidekick on the Late Late Show with Craig Ferguson (**www.cbs.com/shows/late_late_show**). What you may not know is that Geoff is the plastic and aluminum creation of Grant Imahara, a University of Southern California engineering grad, former employee of visual effects company Industrial Light & Magic, and MythBusters guru. The bot started as a joke on Twitter, where Ferguson referred to his followers as a "robot skeleton army." Imahara sent him a tweet offering to build a robot sidekick if the host could help boost Imahara's number of followers past 100,000. Twenty four hours later, it happened. So, Imahara got to work, and the bot debuted on April 5, 2010. Originally, Geoff's voice consisted of only a few prerecorded phrases, but he has been upgraded both mechanically (he can now keep his mouth open, turn his head, and move his right hand) and vocally, with comedian/impressionist Josh Robert Thompson providing live conversation. Thompson has described the character as "one part Snagglepuss, one part Vincent Price, and two parts George Takei." If you're a present or prospective fan, note that you can follow him on Twitter. Just search for GeoffTheRobot and hit the follow button. Soon you'll be receiving tweets like, "I'm a heterojunction bipolar transistor, baby! Translation: I go both ways."

## UAV Tops for Cops


*AeroVironment's Qube Small Unmanned Aircraft System, designed for police and other first responders.*

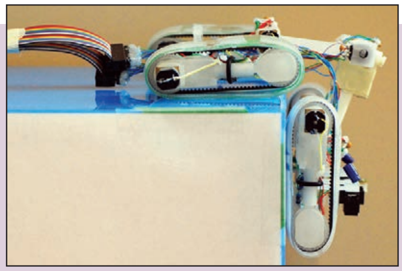
It's not uncommon to see the cops closing in on a fugitive from aloft, often in something like a Robinson R44 police helicopter. That provides a lot of dramatic shots for television, but it's pretty dangerous for the pilot if the perp decides to shoot at him. Plus, it's not exactly a cheap law enforcement tool. The R44 starts at almost $700,000 (which doesn't even include the $2,600 siren), and you'll be running up $193/hr in operating costs. The price has not been released yet, but the new Qube Small Unmanned Aircraft System from AeroVironment (**www.avinc.com**) will accomplish pretty much the same thing "at a fraction of the cost." Specifically tailored to law enforcement, first response, and other public safety missions, it's based on the company's extensive experience with battlefield UAVs. Small enough to fit in the trunk of a black-and-white, it can be unpacked, assembled, and launched in less than five minutes. The company also isn't providing much in the way of performance specs, but it was revealed that the little chopper is equipped with both color and thermal video cameras that can transmit live video to the operator, and it can hover for up to 40 minutes — said to be double the endurance of similar UAVs. In coming months, Qubes will be made available to "select public safety agencies" for testing and evaluation.

## Robotics Rodeo III

If you're a fan of heavy-duty, expensive, and potentially fatal robotic systems, there's great news. The US Army will hold its third Robotics Rodeo in May 2012 at Ft. Benning (near Columbus, GA), and you're invited. This follows successful events in 2009 and 2010. (It was called off in 2011 to allow "a little more time for technologies to mature.")

Robotics Rodeo III — co-hosted by the US Army Tank Automotive Research, Development, and Engineering Center (TARDEC) and Fort Benning's Maneuver Battle Lab — consists of two programs: the "Extravaganza" which is open to the public, and the "Robotic Technology Observation, Demonstration, and Discussion (RTOD2)" which is not. Continuous updates and registration details are available at **www.tardec.info/roboticsrodeo**.

## Half Tank, Half Gecko



*TBCP-11 — a climbing bot from Simon Fraser University.*

Addressing the never-ending need for more agile robots is the Tailless Timing Belt Climbing Platform (TBCP-11), a recent creation at Simon Fraser University (**www.sfu.ca**) up in British Columbia. The prototype has the ability to scale walls with tank-like moves using an adhesive to mimic the sticky toes of a gecko. The idea is to provide "an alternative to using magnets, suction cups, or claws which typically fail at climbing smooth surfaces like glass or plastic. It also paves the way for a range of applications, from inspecting pipes, buildings, airplanes, and even nuclear power plants to employment in search and rescue operations." Unlike a gecko, however, the TBCP uses biomimetic dry adhesives that use Van der Waals forces for adhesion. These are composed of microscale fibers that conform to relatively rough surfaces to maintain the stickiness. If sensors detect a pending detachment, the robot can adjust itself to compensate. At present, the device is said to function "fairly independently," but fully autonomous function remains under development. SV

# GEEK HEAD

by David Geer

Contact the author at geercom@windstream.net

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

# Scout UAV by Aeryon

**The new Scout tactical aerial intelligence UAV is a silent, diminutive flying robot with four vertically facing propellers. The Scout is easily disassembled (don't worry Johnny Five), transported, and reassembled for use. The robot affords aerial intelligence gathering via its onboard camera which operators can use to take video or still pictures. Flight is automated and autonomous via flight plan software and onboard computing using several proprietary processors, freeing the operator's hands, eyes, and mind to focus on gathering images or video.**

Let's examine the robot's design and capabilities, as well as its use in a critical application in one of the world's most important ecological environments.

Greg Walker, Program Manager at the University of Alaska's Unmanned Aircraft Program, uses the Scout, camera, and sensor payloads to examine shorelines and terrain before and after oil spills along Alaska's Arctic coast. Walker and crew take a baseline of the shorelines before potential oil spills happen, then image the shoreline after to note the differences and direct clean-up efforts.

## Flight + Sight = Might

The Scout has a number of capabilities including vertical takeoff and landing (VTOL), flying in any direction including up and down, vertical, left and right, forward and backward, and any combination of these. "It can reliably fly as low as one to two meters in altitude for close-up camera work. It flies in winds up to its maximum speed of 50 km/hr," Walker touts.

The utility that makes its airborne capabilities worthwhile is the fact that it can see, stream, and record what it views through a camera. Manipulating or yawing the camera is done by flying the aircraft sideways rather than by yawing the camera to look sideways. "This is hidden from the operator as you direct where the camera is to point and the direction the aircraft is to fly independent of each other," Walker notes.

The Scout UAV sitting peacefully on rocky ground near the Alaskan coast.

The touchscreen interface on the tablet enables the user to tell the Scout where to go — to an exact position — and point the camera at the desired subject, according to Aeryon. The camera uses GPS targeting so the user can select the position on the ground for the camera to center on during flight.

The AutoGrid software that is utilized enables the user to select the area the camera must cover, and then takes the pictures or video in such a manner that they overlap, therefore imaging all the desired area. This makes GIS (Geographic Information Systems) analysis possible by taking a complete image. All images collected by the Scout are geo-tagged and stored within the system. The information is downloaded to the user system for further processing. Aeryon does not provide GIS tools, but provides the data for use by them.

The Scout's flight system establishes the flight plan that is necessary to take the required images. "At that point, all

the user has to do is launch the UAV, fly up to the necessary altitude, and press go. The Scout then takes the images automatically," explained an Aeryon spokesperson. The flight planning is built into the Scout system and it generates the plan through the touchscreen interface by simply generating waypoints on the map and then saving a file with those in it for later use.

The touchscreen uses satellite map based navigation, meaning that satellite maps of the area are loaded onto the tablet, and the user navigates the Scout and camera based on these. This enables the Scout to achieve an exact location. "The maps are also used to establish the pre-programmed flight paths," according to a spokesperson.

The operator manipulates the touchscreen using a stylus pen. "A click on the map moves the aircraft at predefined maximum speeds towards that point on the map. Lifting the stylus stops the motion of the craft," says Walker. If the operator first selects a button to point the camera, the next stylus depression on the screen's map moves the camera's center of view to that location, leaving the aircraft at the original location, Walker explains.

Other options including zoom, takeoff, and come home are easily identified with the stylus pen. "Only when trying to enter numbers or letters is the stylus interface frustrating," says Walker. "Fortunately, that use of the stylus is quite infrequent."

## Flight Limitations, Communications

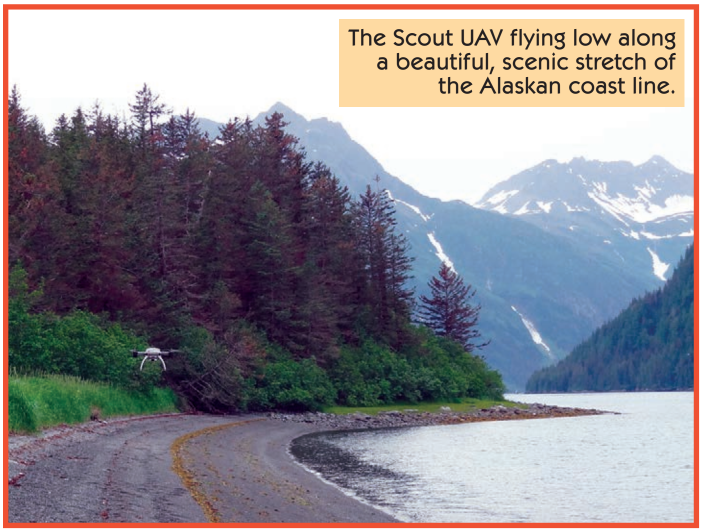
The Scout is designed to operate within the airspace, speed, and altitude restrictions of the given airspace that it is operating in at any given time. These features are integrated into the Scout UAV. This includes an altitude of 500 M, a speed of 21.7 MPH, and airspace restrictions that are determined from country to country. "The operator sets the requirements they need to meet in their airspace. The system uses this data to avoid exceeding airspace area limitations," comments an Aeryon spokesperson. It cannot go beyond these settings. "For example, in Canada the Scout is restricted to airspace under 400 feet. All systems are automatically set to a maximum height of 400 feet. The vehicle will stop at 400 feet even if the operator tries to fly higher."

The Scout comes with its own digital communications network called ScoutNet. Commands and video traverse this network in streams. With the given network design, the operators can stream video to more than one location for viewing purposes. "The network can also be encrypted, ensuring that even if an outside source knew the network address, the information could not be used by an unauthorized party."

Other advantages include multiple user scenarios where operators in a common area can operate the Scout without having to worry about interfacing with each other's controls and data.

"We can monitor the Scout from multiple control stations, and there is a WiFi link from the ground station to any device we want. However, as more systems are linked into the aircraft — such as a second tablet for monitoring the imagery — the network bandwidth gets slower and that is bothersome," says Walker.

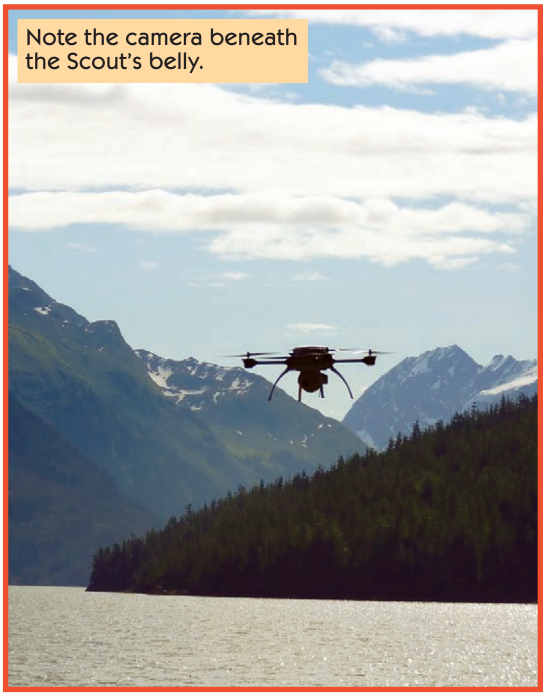
The Scout system complies with STANAG 4586 which is a NATO agreement that defines standard communications protocols, data elements, and message formats for Unmanned Systems. This standard


The Scout UAV flying low along a beautiful, scenic stretch of the Alaskan coast line.


The Scout climbs high above water craft.

### All-Weather Airborne Wonder

In one unique feature, Aeryon has made the Scout weatherproof through sealed seams that prevent moisture and dirt from getting into the robot's system. "While the motors are exposed on the arms," comments an Aeryon spokesperson, " they are designed to meet the same requirements of preventing soil or moisture from entering in." The Scout's payloads are also sealed for weatherproofing.


Note the camera beneath the Scout's belly.


Walker with ground station in hand next to the Scout, readying for takeoff.

was developed in order to meet the goals of interoperability between UAVs. The Scout uses add-on software so that its interface can meet the formats required by STANAG.

## Quick Operator Uptake, Flight Within Minutes

Aeryon provides a three day training course that gives the user all the information necessary to operate the system and all of its features. "We can have a user up and flying in less than 10 minutes," says Aeryon.

An operator can begin to use the Scout immediately when it is first shipped. "From opening the box to turning on the power is about one minute. It then takes the system a few minutes to get the tablet turned on and the program loaded on it. I have consistently been able to put the Scout in the air within five minutes of arriving at a new location. It takes about a minute between battery swaps for the system to self-check," Walker explains. For pre-flight checks, the system verifies all the sensors are operating correctly and ensures a good GPS lock before allowing the user to take off.

A built-in flight simulator gives the user a chance to experience the process of getting the Scout operating on a map. "There is no image on the screen during the simulation, so it is really more of a tool for flight operations familiarity. It was handy for this purpose but was used little after the first day or so of practice," Walker notes.


Walker making notes on the tablet.

The Scout is as mobile as it is maneuverable. An operator can easily remove the robot's arms, legs, payloads, and batteries; place it in its carrying case (which meets commercial passenger plane carry-on requirements in size) along with the operator's touchscreen tablet device, and transport it anywhere.

## Final Thought

The Scout is clearly a very capable advancement in practical unmanned airborne robotics. SV

## Resources

Aeryon, the Scout's makers
**www.aeryon.com**

Aeryon Labs' YouTube channel
**www.youtube.com/aeryonlabs**


What's New?
DARwIn-OP consists of
20 * DYNAMIXEL MX-28,
the same model for all joints.
For more info, please visit
www.RobotSource.org
DYNAMIXEL
MX-Series
MX-28
MX-64
MX-106
S/W Tools
MATLAB®
LabVIEW
Microsoft Visual Studio
python
C/C++ C# Visual Basic
Java
Contactless Absolute Encoder
12-bit resolution, 360° operating range with super-durability
PID control
User Programmable PID Control Gain
Precise and reliable PID control
360° 4096 steps
Powerful Resource & Organized Design
Maxon motor, 32-bit controller, high communication speed
RX and EX series compatible dimension
Torque
Current Based Torque Control
Position and speed control with dual-loop current control (MX-64, MX-106 only)
Usage
[KOREA] E-mail : korea@robotis.com Tel : +82-70-8671-2604
[USA] E-mail : america@robotis.com Tel : +1-949-333-3635
[JAPAN] E-mail : japan@robotis.com Tel : +81-3-4330-3660
[Other Countries] E-mail : contactus2@robotis.com Tel : +82-70-8671-2609
ROBOTIS
www.robotis.com
© OLLO, BIOLOID, and DYNAMIXEL are registered trademarks of ROBOTIS CO., LTD.

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Our resident expert on all things robotic is merely an email away.
**roboto@servomagazine.com**

**Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?**

by Dennis Clark

# ASK MR. ROBOTO

*It's 2012, a new year, with more to learn and more experiments to try. What new wonders will be made available by what seems to be an ever-expanding industry of robotics components suppliers and creators? I'm sure that I don't know, but I'm just as sure I'm going to try to find out! I hope to hear from those of you who try experimenting with these new wonders and come up with questions about new ways to apply them.*

*This month, I'm going to finish up my generic discussion of timers and what we can do with them. The topic will be the PIC microcontroller and its various ways to deal with timers. Each PIC family uses timers; as the PIC has matured, the timer subsystems have changed a little and become simpler and easier to use, in my opinion. Older PIC16C parts made you set a period register, then set your match number, THEN set your counter. Most folks found that very confusing. Newer 16F parts just have a timer pre-scaler setting and the timer counter register (two registers for the 16-bit timer/counters). I'm going to start with the venerable PIC16F series, move to the PIC18F series, and end up with the PIC24 series. You should see the similarities between the timer modules of these very different PIC series variants.*

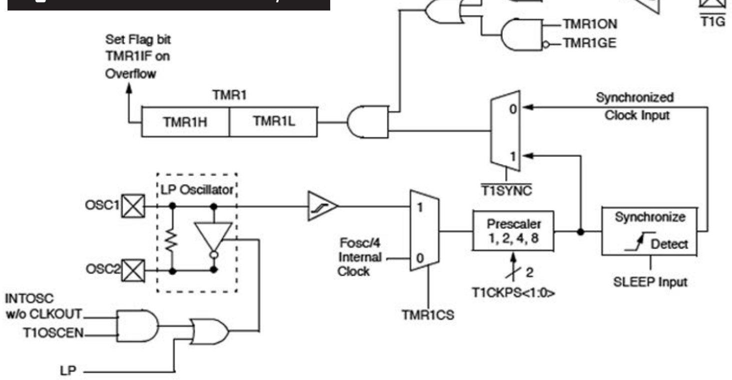
*Before I start though, I'm going to stand on my soapbox for a minute. I still from folks wanting advice with programming 16F<something> parts as a robot controller. My advice to them is "You need to move on!" For all but the simplest robot projects, these parts just won't provide enough "bang for the buck" for you! Even the PIC18F parts should be passed over for the newer PIC24 or even PIC32 devices. These new 16-bit parts are under $4 usually and the 32-bit ones are typically under $7. Microchip has free compilers and a new IDE that runs under Windows, Mac, and Linux operating systems. There isn't any reason not to step up! 'Nuff said.*

*I'm going to choose a C compiler for each of the PIC variants rather than Assembly. While Assembly is fun and all, it isn't the easiest way for newcomers to the craft to use an embedded processor like the PIC. For the PIC16 series, I'll use the CCS PCM compiler. It is a low cost compiler that will integrate with the MPLAB PIC IDE. For the PIC18F series, I'll use the Microchip C18 compiler which can be used by the hobbyist or student for free. For the PIC24F series, I'll use the C24 (also called C30) compiler which is also free for hobbyists and students.*

## Timers for the PIC16/Fxxxx

PIC16F parts typically top out at 20 MHz. The clock to the timers is then Fosc/4, which means when using a 20 MHz crystal your input clock to the timer is 5 MHz. Remember that when you are setting the prescaler to get the timing you want. **Figure 1** shows the hardware layout for the TIMER1 module in a PIC16F630; a very useful 16F series PIC. (*All datasheet graphics used by permission of Microchip, Inc.*)

**Figure 1.** PIC16F TIMER1 layout.




The easiest way to use a timer is to set an interrupt on the *overflow* of the timer's counter register. The timer register will overflow when the register increments from its maximum value (0xFF for an eight-bit timer; 0xFFFF for a 16-bit; and so on) to zero, where it starts counting all over again. To set the interval of that interrupt, you follow this procedure:

**1**. Choose your interval (1 ms, 10 ms, whatever).

2. Choose your prescale to come "close" to your interval upon overflow.
3. Calculate the timer count that comes as close as you can to your interval.
4. Set your timer register to (maximum value – timer count).
5. Code the timer setup and its interrupt.
6. When you get your interrupt, clear it and re-set the register to your value.

Obviously, this process could use an example. Let's set up a background timer on a PIC16F630 to provide us with a 100 ms tic for our timing needs. Let's also say that we're using a 10 MHz crystal on our processor, so our clock to the timer = 10 MHz/4 = 2.5 MHz. This would give us a 400 ns clock which on the 16-bit TIMER1 module would be a 65536 * 400 ns = 26.214 ms rollover period. Hmm. We need 100 ms, so we're going to need to pre-scale the clock in to slow that counter down. How much can we slow it down?

Let's look at the *T1CON* (TIMER1 Configuration) register for our options (**Figure 2**). We see that we only have four choices for a prescaler: 1:1, 1:2, 1:4, and 1:8. Since 1:1 is almost four times faster than what we want, we'll choose 1:4 as our prescale. This takes our clock rate down to 1.6 µS per count with a maximum overflow of 65536 * 1.6 µS = 104.86 ms. That is pretty close to what we want. You'll note that there are a bunch of other options we can set for this timer but we're not using them, so the rest (except for bit 0: TMR1ON) will be left at 0. More on that later when I show some code.

A full timer count is now 104.86 ms which is more than the 100 ms that we want. So, we need to load up the timer register with some value other than zero to come closer. The easiest way to figure out what this number needs to be is by doing some simple math; 100 ms/1.6 µs = 62,500. Our timer register counts up, so that isn't the number we want to "pre-load" it with. We want to use 65536-62500 = 3036. Those were steps 1-4; now we need to write code. **Listing 1** shows how we would set this all up with CCS PCM. PCM will handle clearing the interrupt flag and setting all of the configuration registers for the timer and the interrupt systems. That wasn't painful, was it? Make sure that *t_100ms* is a globally defined variable that is at least 16-bit, preferably 32 bits.

## Timers for the PIC18Fxxxx

The PIC18F series has a higher system clock (40 MHz) and is capable of having much more Flash and RAM memory, making them useful for larger projects. They integrate complex hardware modules like ETHERNET and CAN in some of them, as well. I'm going to refer to the 18F252 microcontroller which is a 28-pin part.

The procedure for figuring out our timer needs is the same as the one given in the previous PIC16Fxxxx series explanation. Let's look at TIMER0 one this part and see what may be different. I'm going to use the Microchip C compiler for the PIC18F series (C18) for the source code example.

**REGISTER 5-1: T1CON — TIMER1 CONTROL REGISTER (ADDRESS: 10h)**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | TMR1GE | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| bit 7 | | | | | | | bit 0 |

bit 7 **Unimplemented:** Read as '0'

bit 6 **TMR1GE:** Timer1 Gate Enable bit
If TMR1ON = 0:
This bit is ignored
If TMR1ON = 1:
1 = Timer1 is on if $\overline{\text{T1G}}$ pin is low
0 = Timer1 is on

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale Value
10 = 1:4 Prescale Value
01 = 1:2 Prescale Value
00 = 1:1 Prescale Value

bit 3 **T1OSCEN:** LP Oscillator Enable Control bit
If INTOSC without CLKOUT oscillator is active:
1 = LP oscillator is enabled for Timer1 clock
0 = LP oscillator is off
Else:
This bit is ignored

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Control bit
TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from T1OSO/T1CKI pin (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

**Figure 2.** T1CON register for TIMER1.

**Figure 3** shows the layout of *T0CON*. You'll note that this TIMER0 configuration register looks a lot like the PIC16Fxxxx T1CON. However, this timer can be configured as either an 8-bit or a 16-bit timer. I chose 16-bit here to get a higher resolution on the timer. Also, TIMER0 has more prescale options on the PIC18F than we had on the PIC16F series.

The Microchip C18 compiler does not abstract the hardware like the CCS PCM compiler does. The programmer has a lot more control over how the interrupts work, but by getting that control, has to do a lot more work setting bits in the configuration registers. **Listing 2** shows what we need to do to set this timer up and configure the interrupt handler. An interesting enhancement to the PIC18F architecture exists in the interrupt module. The programmer can use the *Fast Interrupt* option which has a one-deep stack that holds essential information when an interrupt

**Listing 1: Setup and ISR for TIMER1 overflow.**

```
#INT_TIMER1
void isr(void)
{
     t_100ms++;
     set_timer1(3036);
}

void init(void)
/*
 * sets up everything before I use it.
 * 10MHz clock, external ~MCLR
 */
{
     //Turn off comparator
     setup_comparator(NC_NC_NC_NC);

     //Set up TMR1
     setup_timer_1(T1_INTERNAL | T1_DIV_BY_4);
     //1.6us per tic

     t_100ms = 0;              //set up the ticker
     enable_interrupts(INT_TIMER1);
     //enable TIMER1 interrupt
     enable_interrupts(GLOBAL);
     //turn on the interrupts
}
```

```
#define COUNTER_10MS 59285    // 10ms @ 40MHz
Int2chars       t_timer;      // union converts
                              // 16 bit to 2 8's


#pragma code InterruptHigh = 0x08
               // High Priority Interrupt Handler
void InterruptVectorHigh(void)
               // lives @ 0x08
{
     // jump to interrupt routine
     _asm goto InterruptHandlerHigh   _endasm
}
#pragma code
#pragma interrupt InterruptHandlerHigh
     // INTERRUPT HANDLER
void InterruptHandlerHigh()
{
    //—— Main Timer Tick ———-


     if(INTCONbits.TMR0IF)
       // check for TMR0 overflow
     {
       TMR0H = t_timer.tb[1];
                      // High byte to Timer0
       TMR0L = t_timer.tb[0];
                      // Low byte to Timer0

       t_10ms++;      // 10ms base timer tick
       INTCONbits.TMR0IF = 0;
                      // clear interrupt flag
     }
}
```

```
void Initialize()
/*   Initialize()
     Set up config and control registers in the
     18F252 for future use.
*/
{
     t_timer.val =
     //Interrupt settings
     INTCON   = 0;       //Turn off all interrupts
                         //for now
     INTCON2  = 0;
     INTCON3  = 0;
     RCONbits.IPEN = 1;
               // enable priority levels on reset

     //Main system timer, gets 10ms timer tics
     T0CON = 0x83;       // 16 bit, prescale by 16
                         // for TMR0
     TMR0H = 0;          // count 625 = 1ms
     TMR0L = 0;
     t_timer.val = COUNTER_10MS;
                         // timer setting
     INTCONbits.TMR0IE = 1;
                         // Enable Timer 0 interrupts
     INTCONbits.TMR0IF = 0;
     INTCON2bits.TMR0IP = 1;
               // HIGH priority (low stack overhead)
     CMCON   = 0b00000111;     // Comparators off

     INTCONbits.GIE   = 1;     // Enable global
                               // interrupts
}
```

**Listing 2: C18 TIMER0 setup code.**

occurs. This means that a careful programmer will know exactly what variables are being changed in the ISR so that the whole variable stack does not need to be saved and retrieved during an interrupt. However, you can only have ONE high priority (fast) interrupt happen at a time or you will overflow the stack. This option saves a lot of time, if you are VERY careful!

The first thing you'll notice about the Microchip C18 code is that it is way more complex. Those *pragma* declarations are not easy to understand; you have to know where in memory the interrupts are configured in the hardware to be, and tell the compiler about them. This is because you can move them around if you are using a bootloader or some other fancy setup in your code.

**REGISTER 10-1: T0CON: TIMER0 CONTROL REGISTER**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

**Figure 3.** PIC18F252 T0CON register.

This example assumes that everything is at default locations. I like to use nice and easy numbers for my setups; splitting a 16-bit number into two 8-bit numbers is something for a computer to handle, not my tired brain. Therefore, I created a *union* type called Int2Chars which is a union of a 16-bit integer and an array of two eight-bit integers, called *t_timer*. This allowed me to use a simple 16-bit value (*COUNTER_10MS*) to set the TIMER0 registers, and allowed me to set each individual 8-bit register by just grabbing the upper or lower byte. It may look funky, but it is way easier to just let the compiler use the bits it wants rather than have to do the math in your head!

Since we are a 40 MHz clock frequency, that Fosc/4 = 10 MHz, we then prescale that by 16 giving us a 1.6 µS clock period; 10 ms / 1.6 µS = 6251. Then, 65536 – 6251 = 59285 after you round out the fractions everywhere. This will give us a 10.002 ms tic. Close enough. Setting up the PIC18F was not so easy as the PIC16F series when using Microchip's C18 compiler. If you were to use the CCS PCH compiler, the setup would look the same as the CCS PCM compiler code for the PIC16F series. CCS abstracts most of the nuts and bolts details away from the programmer, but their ISR (Interrupt Service Routine) routines are not as trim as you can make them with C18. So, there you have a trade-off; ease of use vs. flexibility.

## Timers for the PIC24FJ

The last PIC variant that I'm going to discuss setting timers for is the PIC24F series. In this case, the PIC24FJ64GA002. This is a 32 MHz part, but the PIC24 has a two-cycle instruction clock instead of the four-cycle instruction clock on the PIC16 and PIC18. This means that

the PIC24 at 32 MHz is a 16 MIPS microcontroller, whereas the PIC16 at 20 MHz is a 5 MIPS and the PIC18 at 40 MHz is a 10 MIPS microcontroller. If that matters to you, that is.

The compiler I'm going to use is the C24 compiler which is based upon gcc with Microchip libraries. You should see some similarities to the avr-gcc compiler I used in last month's column. This compiler is free for the low optimization version; you'd have to pay if you want the fancier optimizing compiler version.

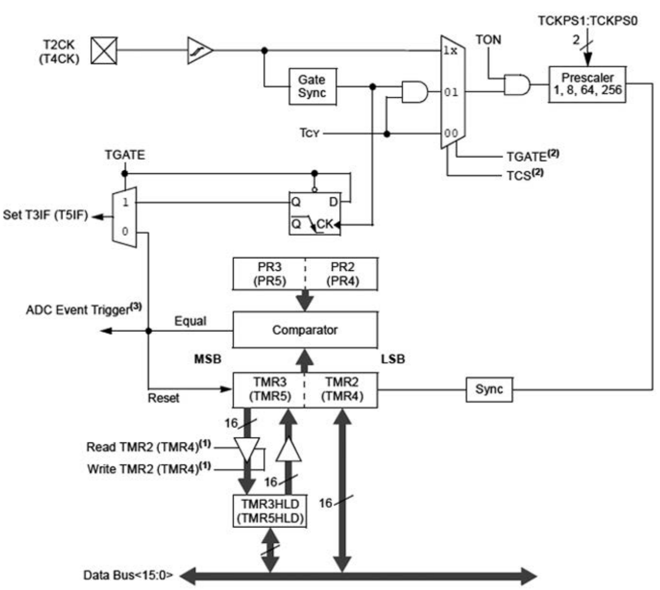
Unlike the other PICs that I've talked about so far, the PIC24F timers are all 16-bit, unless they are 32-bit. Timers 2/3 can be combined; and timers 4/5 can be combined, this is controlled by a bit in their configuration registers. **Figure 4** shows how the TIMER2/3 and TIMER4/5 modules are laid out in the PIC24FJ.

The Tcy input to the timer comes from the system clock; it is Fosc/2. In a full-speed PIC24F system, this would be 32 MHz/2 (the two-cycle instruction clock). Let's look at how to configure TIMER2 (we're going to use the 16-bit timer.) **Figure 5** shows *T2CON* (or *T4CON*), the TIMER2(TIMER4) configuration register. All registers in the PIC24F series are 16-bit, so there are a lot more configuration options here.

Just like in my last examples, I'm going to set up a background timer tic for our program using TIMER2. In this case, it will be a 1 ms ticker. Doing our math to set our period, we see that our input clock is 32 MHz/2 = 16 MHz which gives us a 62.5 ns clock period; 65536 * 62.5 ns = 4.096 ms, which is more than we need. So, we won't need any prescaling. To get 1 ms, we need 1 ms/62.5 ns = 16,000 counts. Here is a place where things just got easier for us! The PIC24F has a period register that we can use to fix the period we want to have a TIMER interrupt from, *PR2*. You can see the *PR2* register in **Figure 5**; between it and the *TMR2* register is a *comparator* which compares the period register with the current count. When these match, a TMR2 interrupt is triggered, the counter register is cleared, and starts counting over again. No mess, no fuss. What a pleasant timer system! In **Listing 3**, I show how to set up the timer and its associated ISR. The PIC24 series has many interrupt vectors built into the system. The PIC16 and PIC18 have just one or two interrupt vectors, and you need to look and see what triggered the interrupt before servicing it. The PIC24 has a unique vector for each interrupt, so you already know what you are doing when the ISR is called.

Note we use volatile variables to hold our timer ticker value. This type of a variable tells the compiler that the value of the variable will change without any program

**Figure 4.** PIC24/fH64/gA002 16/32 timer hardware.




**Note 1:** The 32-Bit Timer Configuration bit, T32, must be set for 32-bit timer/counter operation. All control bits are respective to the T2CON and T4CON registers.
**2:** This peripheral's inputs must be assigned to an available RPn pin before use. Please see **Section 10.4 "Peripheral Pin Select"** for more information.
**3:** The ADC event trigger is available only on Timer2/3.

```
#define         T1MS_SETUP     16000

//Our background timer tickers
extern  volatile uint32_t      t_1ms;
extern  volatile uint32_t      t_10ms;

void Timer2Init(void)
/**
*    Set up Timer2 to provide the 1ms and 10ms
*    ticker clocks. This timer is 16 bits, which
*    is good enough resolution for a 1ms interrupt
*    for the needs a 1ms and 10ms timing
*    resolution. All bits == 0 to get what we want.
*
*    INPUT: none
*    OUTPUT: none
*    RETURNS: none
*
*    SIDE EFFECTS: configures the Timer2 registers
*    for a 1ms clock tick interrupt.
*/
{
     T2CON = 0;                // Clear the register
     t_1ms = 0;
     t_10ms = 0;
     PR2 = T1MS_SETUP;
     IFS0bits.T2IF = 0;        // Clear any IRQ flag
     IEC0bits.T2IE = 1;        // Interrupt enable
     T2CONbits.TON = 1;        // Start the timer
}

void Timer2Off(void)
/**
*    Exists to shut off the TMR2 and its IRQ.
*/
{
     IEC0bits.T2IE = 0;     // Interrupt enable off
     T2CONbits.TON = 0;     // Stop the timer
}

/**
*    This is the 16 bit Timer2 interrupt.  It
*    creates the two timers we use: t_1ms (1ms)
*    and t_10ms (10ms).  Takes 3us to service
*    this.  These use the volatile variables
*    t_1ms and t_10ms.
*/
void __attribute__((interrupt,no_auto_psv,shadow))
_T2Interrupt(void)
{
     static uint16_t tick10=0;

     IFS0bits.T2IF = 0;        //Clear the interrupt
     t_1ms++;             //increment the 1ms ticker
     tick10++;
     if (tick10 == 10) {       //increment the 10ms
                               //ticker
        t_10ms++;
        tick10 = 0;
     }
}
```

**Listing 3: PIC24FJ64GA002 TMR2 ISR code.**

control. A volatile variable's memory location will be read *every time* the variable is referenced; the compiler will know that the code needs to do this because the last time the variable was referenced it may not of held its current value. Always do this for variables that change due to an interrupt. The *static* definition *tick10* tells the compiler that the value must be retained even after the function is done and returns. *Static* definitions act just like module global values, but they are local to the function in which they are defined. These *static* and *volatile* variables are an ansi C standard, gcc is the only ansi C compliant compiler I've talked about in this column. That is why you saw it first with this program.

It may take you a little longer to understand the nuances of the gcc compiler with all its attributes and other arcane looking constructs, but its consistency and completeness will make you a happy programmer in the end. If anyone is interested in the PIC24F's *output compare* or PWM capabilities, I'll do that in another column because the PIC24F allows you to map most of its hardware functions to whatever pins you want them to come out on; the OCP (PWM) functions require that mapping. This can get confusing and really is a topic for an entire column. So, if you are interested, drop me an email and I'll consider it for the future.

Well, I've gone and used up a bunch of your time reading Mr. Roboto again! I hope that you've gotten what you came for. As usual, I enjoyed providing it. So, it's a new year and I'll say it again, "Keep sending me those questions and I'll do my best to answer them!" I can be reached at roboto@servomagazine.com. Enjoy your robots! **SV**

**REGISTER 12-1: TxCON: TIMER2 AND TIMER4 CONTROL REGISTER**

| R/W-0 | U-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| TON | — | TSIDL | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | TGATE | TCKPS1 | TCKPS0 | T32(1) | — | TCS(2) | — |
| bit 7 | | | | | | | bit 0 |

**Legend:**
R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0'
-n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown

bit 15 **TON:** Timerx On bit
When TxCON<3> = 1:
1 = Starts 32-bit Timerx/y
0 = Stops 32-bit Timerx/y
When TxCON<3> = 0:
1 = Starts 16-bit Timerx
0 = Stops 16-bit Timerx

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timerx Gated Time Accumulation Enable bit
When TCS = 1:
This bit is ignored.
When TCS = 0:
1 = Gated time accumulation enabled
0 = Gated time accumulation disabled

bit 5-4 **TCKPS1:TCKPS0:** Timerx Input Clock Prescale Select bits
11 = 1:256
10 = 1:64
01 = 1:8
00 = 1:1

bit 3 **T32:** 32-Bit Timer Mode Select bit(1)
1 = Timerx and Timery form a single 32-bit timer
0 = Timerx and Timery act as two 16-bit timers
In 32-bit mode, T3CON control bits do not affect 32-bit timer operation.

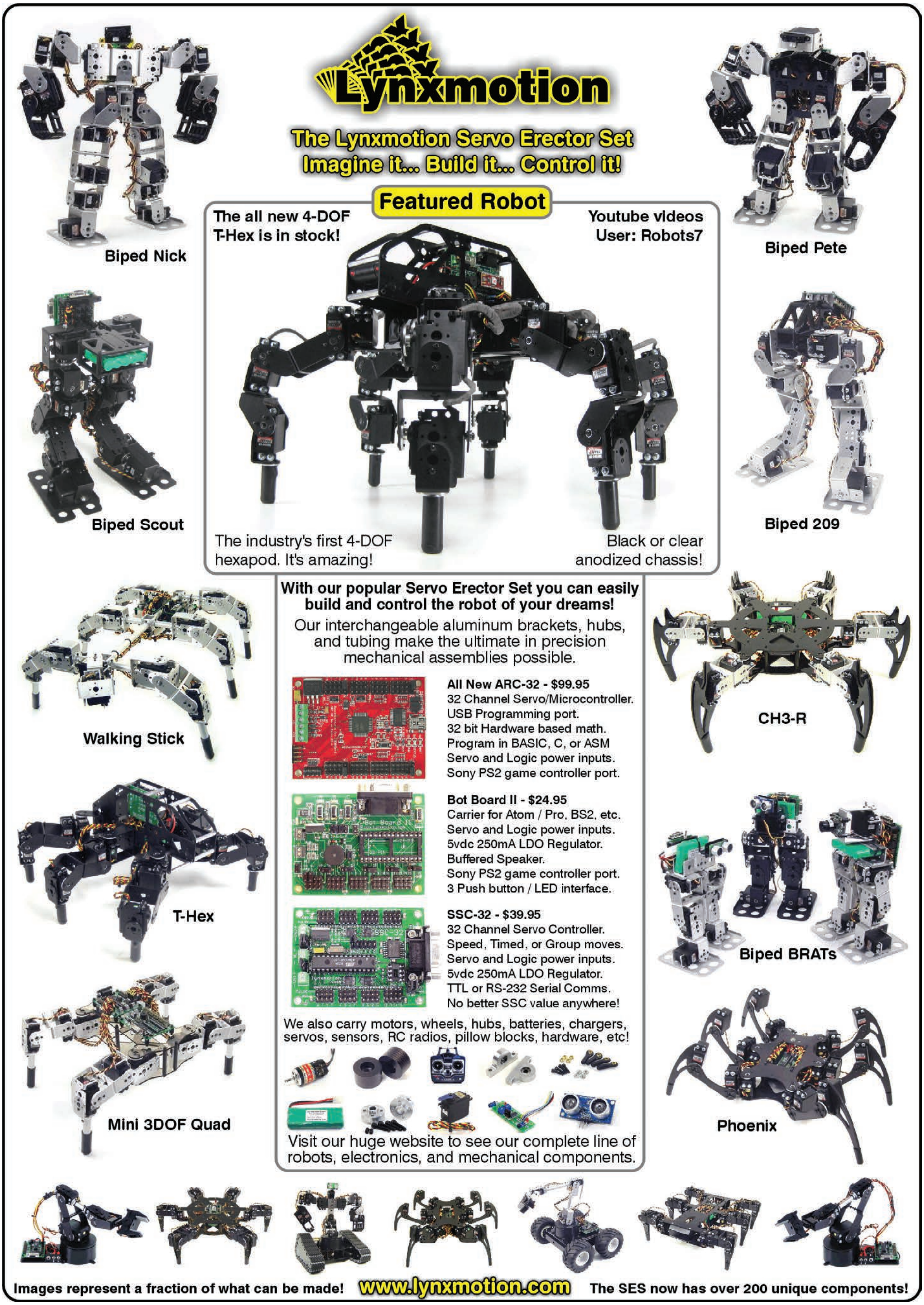
bit 2 **Unimplemented:** Read as '0'

bit 1 **TCS:** Timerx Clock Source Select bit(2)
1 = External clock from pin, TxCK (on the rising edge)
0 = Internal clock (Fosc/2)

bit 0 **Unimplemented:** Read as '0'

**Note 1:** In 32-bit mode, the T3CON or T5CON control bits do not affect 32-bit timer operation.
**2:** If TCS = 1, RPINRx (TxCK) must be configured to an available RPn pin. For more information, see **Section 10.4 "Peripheral Pin Select"**.

**Figure 5.** TCON, TIMER2 configuration register.


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Featured Robot
The all new 4-DOF T-Hex is in stock!
Youtube videos User: Robots7
The industry's first 4-DOF hexapod. It's amazing!
Black or clear anodized chassis!
Biped Nick
Biped Pete
Biped Scout
Biped 209
With our popular Servo Erector Set you can easily build and control the robot of your dreams!
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.
All New ARC-32 - $99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Visit our huge website to see our complete line of robots, electronics, and mechanical components.
Walking Stick
CH3-R
T-Hex
Biped BRATs
Mini 3DOF Quad
Phoenix
Images represent a fraction of what can be made!
www.lynxmotion.com
The SES now has over 200 unique components!

# EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

**— R. Steven Rainwater**

## JANUARY

**6-8** **TECHFEST**
*Indian Institute of Technology, Bombay, India*
This year's contests for autonomous robots include ArchiTech, Autobots, Nexus, Robowars, and Split Second.
**www.techfest.org**

**13-14** **FIRST LEGO League of Central Europe**
*Brandenburg University of Technology Cottbus, Germany*
The Central Europe regional for the FIRST LEGO League — a robot competition for students ages 10 to 16.
**www.hands-on-technology.de/en/firstlegoleague**

**24** **Powered by Sun**
*Ostrava, Czech Republic*
A race of solar-powered autonomous robots.
**http://napajenisluncem.vsb.cz**

**26-29** **ION Autonomous Snowplow Competition**
*St. Paul, MN*
The folks who run the annual autonomous lawnmower competition bring you this new cold-weather robot event for autonomous snowplow robots.
**www.autosnowplow.com**

**27-30** **Robotix**
*IIT Khargpur, West Bengal, India*
Lots of events for autonomous and remote-control bots including Ballista, Pirate Bay, RAFT, RoboCop, Fugitives, and Negotiators.
**www.robotix.in**

**27-30** **Techkriti RoboGames**
*Indian Institute of Technology, Kanpur, Uttar Pradesh, India*
This year's events are Wild Soccer II, The Isle of Torgua, Robot's Got Talent, Lumos, and Ocean's Fourteen.
**www.techkriti.org/#/competitions/robogames**

**31-FEB 2** **Singapore Robotic Games**
*IITE College East, Simei, Republic of Singapore*
A host of events include Picomouse, Sumo, Robot Soccer, Intelligent Robot Competition, Wall Climbing, Pole Balancing, Underwater Robots, Legged Robot Marathon, Robot Colony, Humanoid Competition, and more.
**http://guppy.mpe.nus.edu.sg/srg**

## FEBRUARY

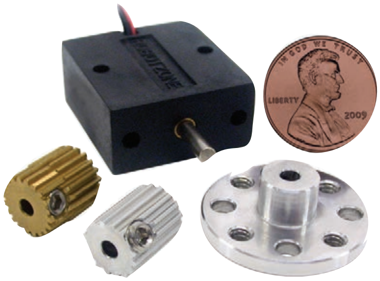
**1-4** **Kurukshetra**
*Guindy, Chennai, India*
Lots of robot events including K!onstructor, Khimera, and Pandemonium.
**www.kurukshetra.org.in**

**5-9** **APEC Micromouse Contest**
*Orlando, FL*
Speedy autonomous mice robots solve mazes.
**www.apec-conf.org**

**23-26** **Pragyan**
*National Institute of Technology, Trichy, India*
Events include Traffic Rush, Fix the Android, Diner Dash, and Lanka Reloaded.
**www.pragyan.org**

# NEW PRODUCTS

## New Micro Gearmotor and Attachments



ServoCity introduces their all new line of micro gearmotors and gearmotor attachments. Although these micro gearmotors have a footprint comparable to a micro-size servo and tip the scales at only 0.6 oz, they are very powerful. All-metal gears ensure the motor will hold up in harsh applications. The versatile gearmotor case provides protection for the gears and offers numerous mounting options to fit a variety of custom applications. These micro gearmotors have a 3 mm D-shaft that protrudes from the gearbox case. The shaft is compatible with ServoCity's new line of 3 mm bore attachments (such as the pinion gears and hub shaft combo pictured). The gearmotors are offered in a variety of RPMs.

## Stacker Robot Kit

ServoCity also introduces the Stacker Robot Kit. This kit is great for both classroom applications and individual robotics enthusiasts. The Stacker wheeled robot kit is machined from 1/4" ABS plastic for strength and durability. The ABS framework also allows for easy drilling to accommodate custom attachments such as cameras, sensors, etc. The framework provides a large electronics compartment that stores the receiver, battery, and any additional added electronics. With 3" drive wheels and omni-directional movement, the Stacker travels well and is easy to maneuver. The Stacker Robot comes in kit form with assembly instructions, a parts list, and a 1:1 scale hardware key for easy assembly. The servos and radio control system are not included with the kit, and must be purchased separately.

For further information, please contact:

| ServoCity | Website: **www.servocity.com** |
| --- | --- |

## Dual Motor Driver Shield for Arduino

Pololu announces the release of the dual VNH5019 motor driver shield for Arduino which provides an easy way to control up to two high power DC motors with an Arduino or Arduino-compatible device. The shield's twin robust VNH5019 motor drivers operate over a wide 5.5V to 24V range and can each deliver a continuous 12A (30A peak) to an independently controlled motor. Or, the driver outputs can be combined to deliver a continuous 24A (60A peak) to a single motor. The motor drivers also offer current-sense feedback and accept ultrasonic PWM frequencies for quiet operation.

This motor driver is intended for a wide range of users, from beginners who just want a plug-and-play motor control solution for their Arduinos (and don't mind a little soldering) to experts who want to directly interface with ST's great motor driver ICs. The Arduino pin mappings can all be customized if the defaults are not convenient, and the VNH5019 control lines are broken out along the left side of the board for general-purpose use without an Arduino. This versatility — along with an option to power the Arduino directly from the shield — sets this board apart. The shield (item #2502) is available for $59.95. For further information, please contact:

| Pololu Corporation | Website: **www.pololu.com** |
| --- | --- |

## Pico Linear Servo

Now available from Solarbotics is the VS-19 pico linear servo which is a tiny and affordable linear actuator with 2 cm of travel. A linear actuator is an actuator that creates linear motion, as opposed to rotary motion, i.e., the spinning of a motor. Many linear actuators (such as this one), are actually driven by a non-linear motion. The VS-19 uses a geared pager motor to spin a worm gear section with a floating nub.

The VS-19 can be used to turn light switches on and off, raise and lower legs, trigger a latch, and/or bring up a periscope. Because of its size, it's perfect for stealthy remote control applications. This actuator accepts a servo pulse from 800-2,200 microseconds (uS) with a neutral position of 1,500 uS. The traveling nub takes about a second to move from one side of the channel to the other (2 cm) at 3.7V. This linear servo can run at voltages as low

*Continued on page 51*

# bots IN BRIEF

## FLIGHT NIGHT

This might look like a state-of-the-art theme park ride, but the thrills with Deakin University's latest innovation will only be experienced by the likes of future jet fighter pilots.

Yep. That's a dude playing around in the immersive flight simulator that's mounted on the end of a giant robot arm. It has six degrees of freedom and it can simulate continuous rotation and g-forces. (Doesn't that sound like some serious puke-inducing fun, or what?) The robot — essentially a heavily-modified industrial arm — is at the center of Deakin University's Universal Motion Simulator (UMS), a facility specifically designed to train fighter pilots. While other simulators can provide some sense of motion, the UMS can generate up to 6 Gs of force which you wouldn't otherwise experience outside of a high speed turn in a fighter jet (or maybe a rocket launch). While this is all going on, the UMS will send back data on the users vital signs to make sure their eyeballs are still in the sockets.

"This next generation simulator uses its oversized robot arm to spin users at high speeds in any direction," explains Professor Saeid Nahavandi, the Director of Deakin's Centre for Intelligent Systems Research (CISR). No other simulator can provide the full experience of flying a military jet with all the gut-wrenching G forces while only seven meters off the ground.

When the Australians get a couple of these things up and running, they'll be linked together to let fighter pilots dogfight with their buddies.

What sets the UMS apart from standard simulators is the integration of haptics technology which provides a sense of touch and feel to virtual or remote objects, and its ability to move at high speed and in any direction. Combined with a high resolution 3D display mounted inside a headset, the user is totally immersed in the set training environment and has a "real" experience – both visually and physically.

**www.deakin.edu.au/news/2011/181111flightsimulator.php**

## WORM YOUR WAY AROUND

There's just something "interesting" about the bulbous air muscles that soft robots use. The designs continue to get more and more refined so the robots themselves are getting more and more capable of actually doing stuff. Take this soft robot from Harvard, for example. It not only walks, it knows several different gaits and can deflate to stuff itself through tiny little gaps.

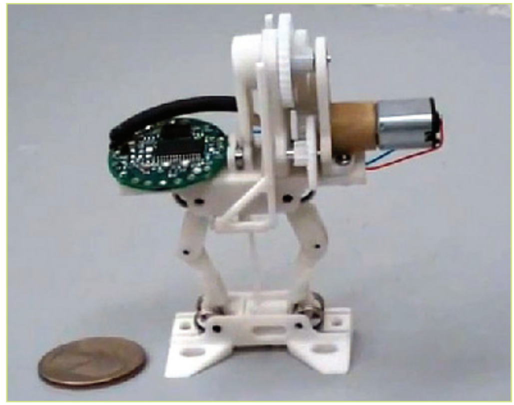
There's nothing solid in it at all. You could probably smash this thing with a hammer a whole bunch of times and it would still keep going. However, that's part of the idea. The other part of the idea is that soft robots can adapt themselves to squeeze through tight places and otherwise get into areas that robots with rigid structures might not be able to.

This particular robot (which comes from George M. Whitesides' lab at Harvard) definitely distinguishes itself by being capable of several unique gait styles including walking, crawling, and slithering. Each of the gaits is controlled by pumping air at up to 10 PSI into a succession of limbs, inflating and deflating elastomer compartments to provide temporary structure and rigidity. In addition to slipping through gaps, the robot can make it across things like felt cloth, gravel, mud, and Jell-O (don't ask how they know).

As the Harvard researchers explained in a paper, the robot was inspired by animals like squid, starfish, and worms that "do not have hard internal skeletons," and the advantage of soft robotics is that "simple types of actuation produce complex motion."

**www.physorg.com/news/2011-11-gumby-like-flexible-robot-tight-spaces.html**

## MADE YOU JUMP!



This robot is easily one of the most brilliant designs we've seen. This little guy was first introduced at the 2010 IEEE International Conference on Robotics and Automation, where it showed off its ability to jump, land without smashing itself to pieces, stand up again, turn, and then make another jump.

The epic cleverness of this robot (which was developed by Jianguo Zhao at Michigan State University) comes from the fact that it uses just one single pager motor to jump, self-right after landing, and then orient itself to make its next jump in the right direction.

It's just gotten some substantial improvements. It has nearly doubled its jumping height to just under a meter, which is about 14 times the height of the robot itself. It can now turn much faster, at 36 degrees per second — up from two degrees per second. Plus, the self-righting system is significantly more robust. All of this stuff has happened without the robot increasing in size or weight — which is fairly remarkable — and it's so efficient that it can jump hundreds of times without needing to recharge.

A robot like this has all sorts of potential uses, although most of them fall into the (familiar) categories of search and rescue, environmental monitoring, and military surveillance. It'll be straightforward enough to mount a payload (like a wireless camera) onto this little dude, but in order to be really useful, it's probably going to have to learn how to right itself on non-flat surfaces. But have no fear! There's bound to be some sort of clever little tweak that'll make this robot able to jump from rough surfaces, clear tall buildings in a single bound, solve the world's energy problems, and play the piano — all on one pager motor.

Alternative locomotion methods such as jumping and flying posses several advantages compared to wheeled locomotion. For example, the jumping sensors can overcome obstacles higher than themselves which is impossible for wheeled ones. Therefore, jumping provides an ideal solution for rugged terrain manueverability.

**www.egr.msu.edu/~zhaojia1/doku.php**

## DOING THE WAVE (GLIDER)

Recently, four Wave Gliders — self-propelled robots, each about the size of a dolphin — left San Francisco, CA for a journey that combined will total 60,000 kilometers. Built by Liquid Robotics, the bots will travel together to Hawaii, then split into pairs. One pair will head to Japan, the other to Australia. Waves will power their propulsion systems and the sun will power the sensors that will be measuring things like water salinity, temperature, clarity, and oxygen content; collecting weather data; and gathering information on wave features and currents. It's not going to be an easy journey — the little robots will face rough weather and have to dodge big ships.

The data from the fleet of robots is being streamed via the Iridium satellite network and is freely available in an accessible form on Google Earth's Ocean Showcase and in a more complete form to researchers who register. Liquid Robotics is eager to see what the scientific community does with all the data. So eager, in fact, that they're asking for project abstracts, and will give a prize to the top five proposals: six months' use of a Wave Glider optimized to collect whatever information the winner needs.

Liquid Robotics' technology is very different. For one, the Wave Glider is a boat, not a submarine. It sits on top of the water. That lets it pick up information about weather, waves, and currents that just aren't available to submarines. It moves much differently, as well. Submarine gliders propel themselves by repeatedly changing their buoyancy; the Wave Gliders use the motion of surface waves to paddle underwater fins. Because most of the body of each craft sits above the water, it gets a lot of sunlight. So, the deck is covered with solar cells that recharge the battery that powers the sensors and transmitters. (Conveniently, the splashing of the waves helps keep the solar panels spotless.) **http://liquidr.com**

## ROACH COACH

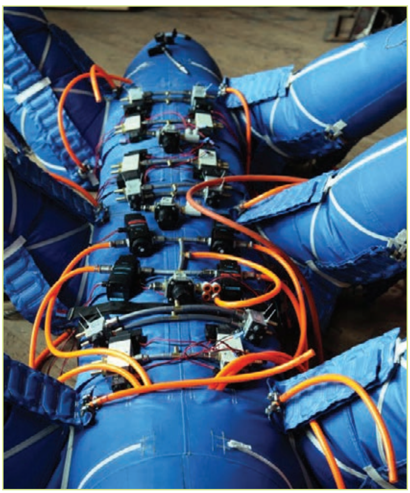
Meet Ant-Roach. Ant-Roach is called Ant-Roach because to those with a fanciful imagination, it looks a bit like a cross between an anteater and a cockroach. However, it'll take an even more fanciful imagination to figure out how to naturally make that combo come to pass. Imagination or no imagination, Ant-Roach exists, it moves, and you can ride it. Plus, it's completely inflatable — muscles and all.

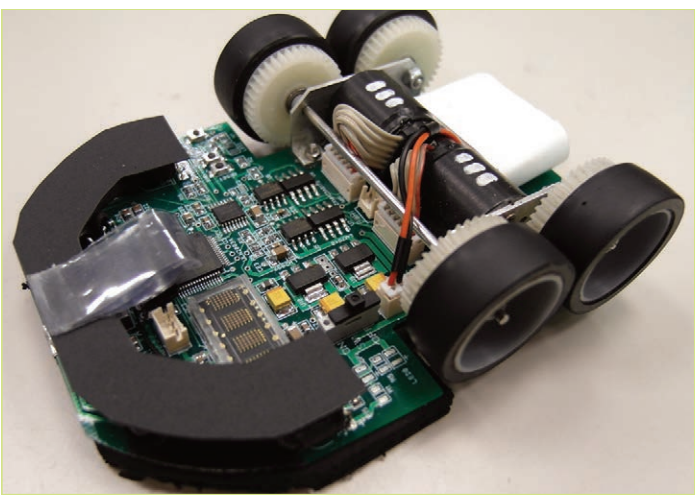
Ant-Roach weighs only 70 pounds (seeing as it's hollow and made of fabric) which means that one reasonably in-shape person can carry it around (as long as it's been deflated first). As you can see from the photo, the robot is capable of supporting a lot of weight — as much as 1,000 pounds. To move, Ant-Roach uses what looks to be four independently controllable pneumatic bags that have been designed to contract when inflated. By attaching these bags to the legs and torso of the robot, and inflating and deflating them in sequence, Ant-Roach can be made to walk, turn, and even swim (well, sort of).

The muscles are driven from several central manifolds which dispense compressed air. A microcontroller receives a wireless signal from a laptop running the control program to drive the robot.

So, why an inflatable robot? There are lots of reasons: they're cheap, they're (relatively) easy to build, they're (also relatively) easy to fix, and they have very high strength-to-weight ratios. Perhaps most importantly, being full of air, inflatable robots tend to be much more compliant than their metallic brethren, meaning that they're inherently safer to have operating around humans.

**www.otherlab.com/news/2011/11/21/the-ant-roach/**





## MIGHTY (FAST) MOUSE

Last year, a micromouse managed to negotiate a maze in under five seconds. At the 2011 All Japan Micromouse Robot Competition in Tsukuba, the micromouse shown here shaved an entire second off of that time, completing the maze in a mere 3.921 seconds. That's fast!

This robot — called Min7.1 — was designed by Ng Beng Kiat. It has a top speed of just over 12 kph which is wicked quick for something that's 10 cm long and weighs only 90 grams. Of course, the micromouse has to figure out where it's going before it can put the hammer down and blaze through on its final run, which is why it first gets an autonomous exploration phase.

**www.np.edu.sg/alpha/nbk/**

*Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.*

# ARDUINO - Simple to Advanced Projects

## ARDUINO DEVELOPMENT KITS

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs (includes Jaycar stepper motors). Arduino projects can be stand-alone, or they can be communicated with software running on your computer. These Arduino development kits are 100% Arduino compatible. Designed in Australia and supported with tutorials and guides. See website for complete Arduino range.

### "Eleven" Arduino-compatible development board

**XC-4210 $29.00 plus postage & packing**

An incredibly versatile programmable board for creating projects. Easily programmed using the free Arduino IDE development environment, and can be connected into your project using a variety of analog and digital inputs and outputs. Accepts expansion shields and can be interfaced with our wide range of sensor, actuator, light, and sound modules.

- ATmega328P MCU running at 16MHz
- 14 digital I/O lines (6 with PWM support)
- 8 analog inputs

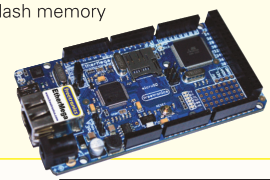
### EtherMega, Mega sized Arduino compatible with Ethernet

**XC-4256 $85.75 plus postage & packing**

The ultimate network-connected Arduino-compatible board: combining an ATmega2560 MCU, onboard Ethernet, a USB-serial converter, a microSD card slot for storing gigabytes of web server content or data, Power-over-Ethernet support, and even an onboard switchmode voltage regulator so it can run on up to 28VDC without overheating.

NEW 2011

- ATmega2560 MCU running at 16MHz, large Flash memory
- 10/100base-T Ethernet built in
- 54 digital I/O lines
- 16 analog inputs
- MicroSD memory card slot
- Prototyping area
- Switchmode power supply



### USBDroid, Arduino-compatible with USB-host support

**XC-4222 $50.50 plus postage & packing**

This special Arduino-compatible board supports the Android Open Accessory Development Kit, which is Google's official platform for designing Android accessories. Plugs straight into your Android device and communicates with it via USB. Includes a built-in phone charger.

- ATmega328P MCU running at 16MHz
- USB host controller chip
- Phone charging circuit built in
- 14 digital I/O lines (6 with PWM support)
- 8 analog inputs
- MicroSD memory card slot

### EtherTen, Arduino-compatible with Ethernet

**XC-4216 $50.50 plus postage & packing**

This Arduino-compatible development board includes onboard Ethernet, a USB-serial converter, a microSD card slot for storing gigabytes of web server content or data, and even Power-over-Ethernet support.

- ATmega328P MCU running at 16MHz
- 10/100base-T Ethernet built in
- Used as a web server, remote monitoring and control, home automation projects
- 14 digital I/O lines (6 with PWM support)
- 8 analog inputs

## Getting Started with Arduino

**BM-7130 $20.75 plus postage & packing**

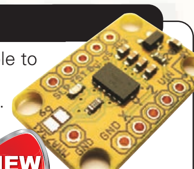
This book explains what Arduino is, how it works and what you can do with it. It also includes a project to build, complete with how to write the code to make it work.

- Softcover, 118 pages. 216 x 140mm


NEW 2011

## Arduino Modules

We have a huge range of simple to advanced add-ons that provide input for your Arduino projects. Visit our website for our full range and more details.



NEW 2011

| | |
|---|---|
| **N-MOSFET Driver & Output Module** | XC-4244 $5.25 |
| **Logic Level Converter Module** | XC-4238 $5.25 |
| **Shift Register Expansion Module** | XC-4240 $5.25 |
| **Light Sensor Module** | XC-4228 $7.25 |
| **Sound & Buzzer** | XC-4232 $7.25 |
| **Microphone Sound Input Module** | XC-4236 $7.25 |
| **Hall Effect Magnetic & Proximity Sensor Module** | XC-4242 $7.25 |
| **Full Colour RGB LED Module** | XC-4234 $7.25 |
| **Temperature Sensor Module** | XC-4230 $12.25 |
| **3-Axis Accelerometer Module** | XC-4226 $14.50 |
| **Humidity & Temperature Sensor Module** | XC-4246 $14.50 |

## ProtoShield Basic

**XC-4214 $3.25 plus postage & packing**

A prototyping shield for the Eleven (XC-4210) and USBDroid (XC-4222) both featured above. Provides plenty of space to add parts to suit any project, keeping everything neat and self-contained. Includes dedicated space to fit a power LED and supply decoupling capacitor.

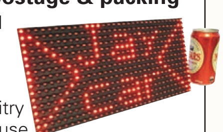
- Gold-plated surface

NEW 2011

## Large Dot Matrix Display Panel

**XC-4250 $29.00 plus postage & packing**

A huge dot matrix LED panel to connect to Eleven, EtherTen and more! This bright 512 LED matrix panel has on-board controller circuitry designed to make it easy to use straight from your board. Clocks, status displays, graphics readouts and all kinds of impressive display projects are ready to create with this display's features.



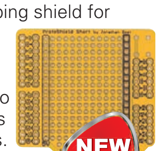
- 32(L) x 16(W)mm high brightness Red LEDs (512 LEDs total) on a 10mm pitch
- 5V operation
- Viewable over 12 metres away
- Tough plastic frame
- Controller IC's on board, simple clocked data interface
- Arduino compatible library, graphics functions and example support

NEW 2011

## ProtoShield Short

**XC-4248 $3.75 plus postage & packing**

A dedicated short version prototyping shield for EtherTen and EtherMega. This special prototyping shield is designed to fit neatly behind the RJ45 Ethernet jack, allowing you to stack your Ethernet-based projects right on top with standard headers.



NEW 2011

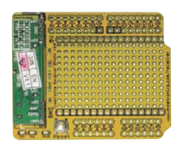
- Pads available to fit a reset button
- Gold-plated surface for maximum durability

## Receiver Shield 433MHz

**XC-4220 $21.75 plus postage & packing**

This receiver shield lets you intercept 433MHz OOK/ASK signals, decoding them in software on your Arduino. All the Arduino headers are broken out to solder pads, and GND and 5V rails are provided for convenience.

- Reset button
- Blue "power" LED
- Red and green user-defined LEDs
- Gold-plated surface
- 433.92MHz tuned frequency



NEW 2011

## LCD & Keypad Shield

**XC-4218 $21.75 plus postage & packing**

Handy 16-character by 2-line display ready to plug straight in to your Arduino, with a software-controllable backlight and 5 buttons for user input. The display is set behind the shield for a low profile appearance and it includes panel mounting screw holes in the corners.

- 2 rows of 16 characters
- Supported by a driver library
- Software-controlled backlight
- Reset button
- Dimensions: 85(W) x 54(H) x 12(D)mm (24mm including header pins)


NEW 2011

## Post & Packing Charges

| Order Value | Cost |
|---|---|
| $25 - $49.99 | $7.50 |
| $50 - $99.99 | $20 |
| $100 - $199.99 | $40 |
| $200 - $499.99 | $60 |
| $500+ | $75 |

Max weight 12lb (5kg).
Heavier parcels POA.
Minimum order $25.

Note: Products are dispatched from Australia, so local customs duty & taxes may apply.

Prices valid until 31/1/2012

## HOW TO ORDER

| | |
|---|---|
| WEB: | www.jaycar.com |
| PHONE: | 1-800-784-0263* |
| FAX: | +61 2 8832 3118* |
| EMAIL: | techstore@jaycar.com |
| POST: | P.O. Box 107, Rydalmere NSW 2116 Australia |

• ALL PRICING IN US DOLLARS • MINIMUM ORDER ONLY $25

*Australian Eastern Standard Time
(Monday - Friday 09.00 to 17.30 GMT + 10 hours)
Note: 10-14 days for air parcel delivery

**Order online: www.jaycar.com**

Jaycar Electronics

# COMBAT ZONE

Discuss this section in the SERVO Magazine forums at http://forum.servomagazine.com

## Featured This Month:



www.servomagazine.com/index.php?/magazine/article/january2012_CombatZone

# PARTS IS PARTS

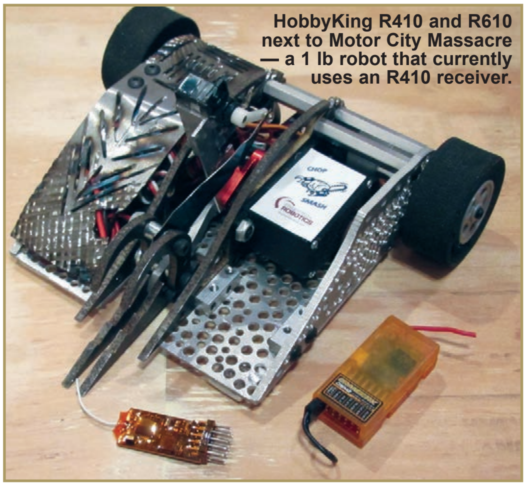
## HobbyKing R610 and R410 2.4 GHz Receivers

● by Mike Jeffries

I found out recently that there are very few DSM2 receivers with programmable failsafes available when I decided to replace the Spektrum BR6000 receivers I had been using with a DSM2. This change lead to a search for a receiver that would failsafe correctly with my DX6i transmitter and a wide range of motor controllers, including the IFI Thor 883, the Holmes Hobby BR-XL, and the FingerTech Robotics Tiny-esc. At the time, I selected the cheapest Spektrum branded receiver that was listed as failsafing correctly — the AR6115.

In testing, this proved not to be the case as the manufacturer had changed the failsafing properties when the model number switched from 6110 to 6115. I next tried the AR600 after returning the AR6115. The failsafe behavior was still not correct.

After investigating the issue, the culprit was found to be the means of failsafing. The BR6000 had programmable failsafes; the AR600 failsafes were enabled by sending a null signal upon signal loss which works for some — but

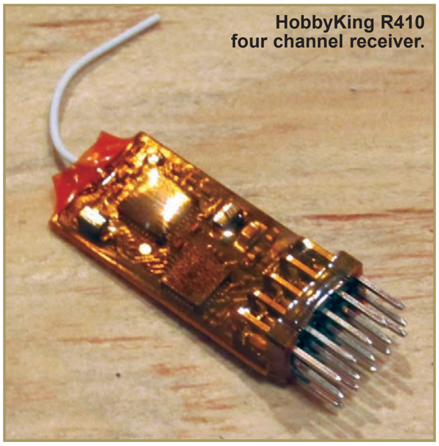

HobbyKing R410 and R610 next to Motor City Massacre — a 1 lb robot that currently uses an R410 receiver.

not all — motor controllers. When selecting a receiver, make sure the type of failsafe used is compatible with the motor controllers you intend to use.

Before moving into the part review, I want to discuss what you need to look for in a failsafe. Failsafing is the idea that if the receiver loses contact with the transmitter, the receiver will send a certain signal to the motor controllers and servos that results in the vehicle acting in a predetermined manner. For RC aircraft, this is often maintaining the last sent signal and going to a preset (normally off) throttle position. For RC cars and some other RC aircraft, it makes more sense to go to neutral positions with the throttle cut off, resulting in a straight path and deceleration.

For fighting robots, you want a failsafe system that results in the weapon and drive systems deactivating, making them safe to approach and power off. Making sure your machine failsafes properly can be the difference between a stationary robot and a missing limb. When setting failsafes, make sure to test with as much of the weapon or drive system disconnected as possible to minimize the risk of an improper failsafe causing damage or injury.

Spektrum currently produces only one receiver that has fully programmable failsafes: the AR6255. With a price of $79.99, it can become a major expense on smaller robots. Spektrum is not the only source for receivers that will work with Spektrum transmitters. HobbyKing has a line of low cost receivers that are designed to work with Spektrum's DSM2 technology.



HobbyKing R410 four channel receiver.


HobbyKing R610 six channel receiver.

Weighing in at under three grams, the four channel HobbyKing R410 is a great option for weaponed robots that are tight on space or weight. With dimensions of only 37 x 15 x 8 mm, this is one of the smallest options available today. In testing, the R410 has never had any obvious connectivity issues and bound as quickly as the official Spektrum receivers.

Switching from the BR6000 to the R410 in my 1 lb robot "Motor City Massacre" saved valuable weight and space in a tightly packed chassis. In addition to performing perfectly in every test, the receiver sells for only $9.99 through **HobbyKing.com**, making it a great low cost option for any combat robot that has little room or weight to spare.

The R610 is the larger six channel receiver made by HobbyKing for Spektrum transmitters. At 9.8 grams and measuring 43 x 22 x 13 mm, this is a fair bit larger than the R410 but performs just as well and comes at an even lower price, only $5.99. As with the R410, I have yet to have a failure with the R610 in testing and have it installed on my 30 lb bar spinner "Moros".

Based on my tests, I've replaced all of my BR6000 receivers with HobbyKing RX10 receivers and will continue to use them in future builds. With performance on par with the BR6000 and the cost per unit being so low, they make a great alternative to the outdated BR6000 and the much more expensive AR6255. SV

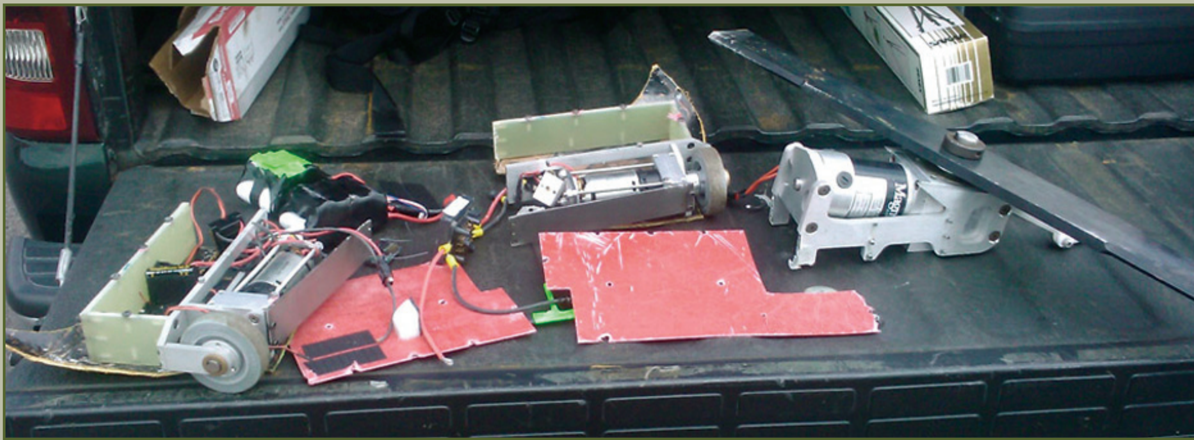
# MANUFACTURING:

## Shop Review: Westar Mfg. – The Team Whyachi Bot Shop

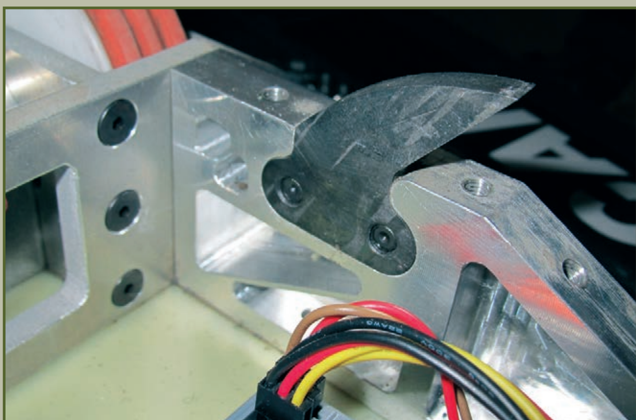
● by Mike Jeffries

Sometimes you just don't have the equipment or the time to make the parts you need right. I often get asked where I get my machining done, and unless I've done it myself, my answer is always **teamwhyachi.com**. The great quality, reasonable prices, wide range of materials, and short lead times have kept me going back since 2004.

The Team Whyachi Bot Shop is a full service machine shop in Dorchester, WI and in addition to custom work, supplies gearboxes,



The custom right angle gearbox from Team Whyachi survived the rest of the robot exploding in testing.



Extremely tight machining tolerances on the second build of Apollyon.

power switches, and high strength wheels to meet a wide range of needs.

The first custom part I got from Westar Mfg. is still in use today. In 2004, I had them make a custom version of their TWM3R spin shaft gearbox for use in my old 60 lb bar spinner "Ruiner." So far, the gearbox has outlasted Ruiner and my first attempt at a 30 lb version ("Mr. Self Destruct") which exploded in testing. It is currently in use in its third robot, "Moros".

The Westar Mfg. shop has never failed to provide high quality parts in a short amount of time. In the second build of my 12 lb wedge "Apollyon", I used an absurd amount of pocketing and slide fits to create a dense, strong chassis. The parts arrived quickly and fit perfectly which resulted in a build that took less than a day once all the parts had been gathered.

In addition to machining, the Team Whyachi Bot Shop offers waterjet cutting services. They have a great machine and can achieve tolerances as low as ± 0.002" while making holes as small as 0.090". The dynamic head allows for parts or stacks of material up to 6" thick to be cut, and with a six ft by 12 ft working area, you'll have plenty of space to work with when designing your parts.

The high quality and fast turnaround has kept me going back to them again and again. I've got parts made at the Team Whyachi Bot Shop in all four of my current active robots and plan on using them for my next project. In my experience, if you need parts done right and done fast, **teamwhyachi.com** is the place to go. SV

Waterjet parts for 1 lb and 30 lb robots from Westar Mfg.

Every robot pictured uses parts made by Westar Mfg.

# BUILD REPORT:

## Rollin' With the Punches

● by Andrea Suarez

Win or lose, the best combat robots always put on a show. That means no drive system failures unless your robot is in pieces. Bigger bots can achieve this through redundancy: four motors, four speed controllers, etc. However, a single motor-controller-wheel combo can be more than 15% of your insect weight bot!

The first revision of WhipperSnapper — my one pound robot with a modular (horizontal or vertical) five inch spinning disk — had been cursed with persistent gearmotor and wheel failures. In planning for a redesign, choosing the right components became essential. I switched from the FingerTech Gold Spark to the new Silver Spark gearmotors with larger pitch gears for vastly improved reliability (still under $25).

The FingerTech Lite hubs are much lighter than other aluminum hubs, and they provide two set screws on each hub. Even other lowest durometer wheels transferred most of the shock to the gearbox and provided little forgiveness during impacts, so I decided to use the much softer foam Lite Flite wheels instead.

This new drive combo has survived its first two full events (13 matches) with no drive system failures, and saved me enough weight for a much needed weapon upgrade!

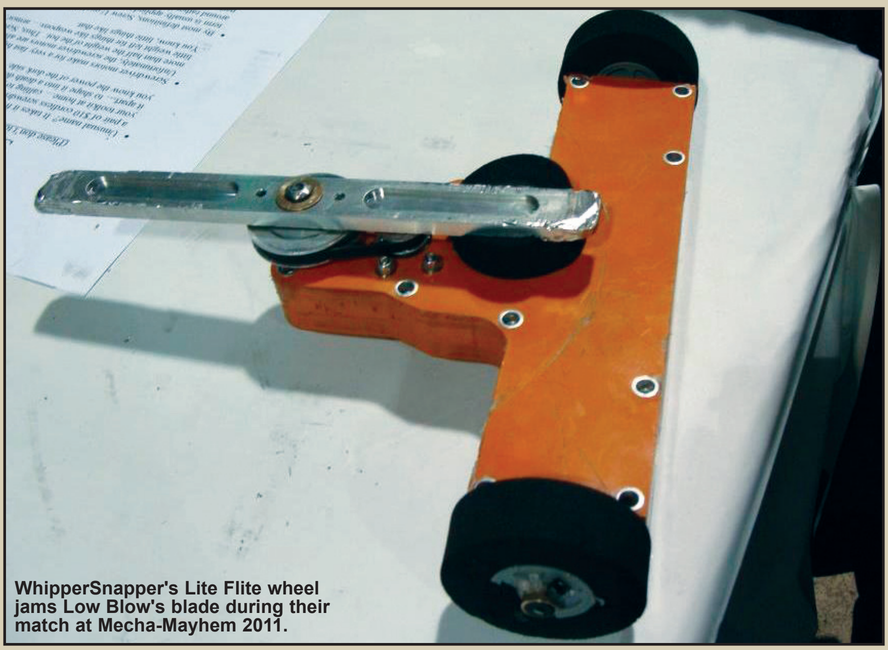
The real test to this new setup, however, came in a Mecha-Mayhem match-up against Low Blow from Pretzel Robotics. Thirty seconds into the match, Low Blow's giant under-cutter got a big hit to WhipperSnapper's wheel, cracking the gray snap-on plastic and pulling off the foam. In a turn of fate, WhipperSnapper's wheel got wedged between Low Blow's spinning bar and the bottom of his frame, stalling the weapon and deflecting his bar far enough to raise both of his wheels off the floor!


Lite Flite wheels after Mecha-Mayhem 2011. No Silver Spark gearbox or motor failures despite an intense event!

My 22.2:1 Silver Sparks survived the hit and the FingerTech Lite hub remained secured to the motor shaft. The damage was easy to repair — a dab of CA glue to hold the foam wheel onto the remaining Lite Flite plastic and WhipperSnapper was back in action.

Having the Lite Flite wheel fail before the FingerTech hub insures that your robot won't lose its wheels unless they are destroyed, saving the embarrassment of seeing your wheel roll away as the announcer starts the countdown. These easy drive changes shot WhipperSnapper's record from 3-2 to 14-4 in just two events, winning 2nd at GCRS 8 and 1st at Mecha-Mayhem 2011. SV


WhipperSnapper at competition after drive system redesign.


WhipperSnapper's Lite Flite wheel jams Low Blow's blade during their match at Mecha-Mayhem 2011.

# EVENT REPORT:

## *Mecha-Mayhem – The Rumble in Rosemont*

● by Dave Graham

Fighting robot enthusiasts from Miami to Michigan and their 47 insect class fighting robots journeyed to the Chicago suburb of Rosement, IL to do battle in the 5th Annual Mecha-Mayhem fighting robot competition. Mecha-Mayhem is the flagship event for the Chicago Robotic Combat Association (CRCA), and features two days of robot destruction in a state-of-the-art arena that sports arena hazards in the form of four pneumatically operated hammers in the arena corners.

**FIGURE 1. Team Busted Nuts (from left to right - Andrea Suarez, Michael Gellatly, Paul Grata, and Jennifer Villa).**

Event organizers Brian Schwartz and Dave Miller partnered with the iHobby Expo to add this combat robot event to the venue. The iHobby Expo is the largest trade and consumer hobby show in the United States, and provides an international showcase for the newest electronic hobbyist technology, models, and valued miniature collectables.

Mecha-Mayhem 2011 drew nearly double the competitors from previous years. Those competitors included several new teams, most notably Team Busted Nuts from Miami, FL (**Figure 1**).The Busted Nuts team members are veterans of the high school Bots IQ competition and are used to building 120 pound fighting monsters. This was their first foray into insect class fighting robots, and their first road trip to a competition. They brought an impressive collection of well-designed, durable Flea (a.k.a., Fairy), Ant, and Beetleweight bots, and established themselves early on as the team to beat.


**FIGURE 2. Washington University ASME Team mega charging station.**

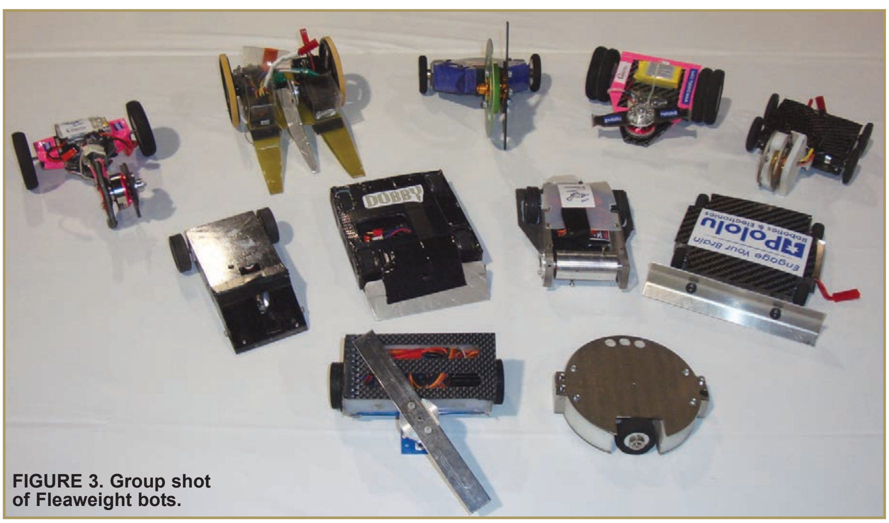


**FIGURE 3. Group shot of Fleaweight bots.**

I need to give a shout-out to Michael Winek from the American Society of Mechanical Engineering (ASME) Washington University team. While Michael didn't attend the competition, he designed an impressive mega charging station (**Figure 2**) for his ASME teammates.

The Fleaweight competitors definitely raised the bar this year, assembling the largest group of bots with the most diverse weaponry I've ever seen. Eleven Fleas (**Figure 3**) sporting horizontal spinners, wedges, vertical spinners, a spinning drum, a lifter, and yes, a full-body spinning Melty Brain bot

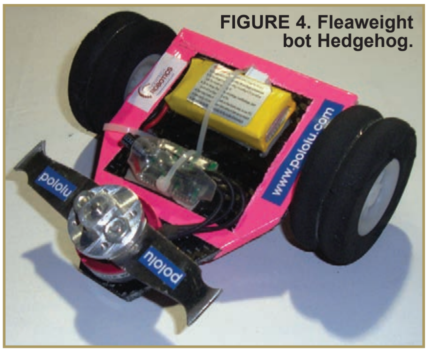


FIGURE 4. Fleaweight bot Hedgehog.

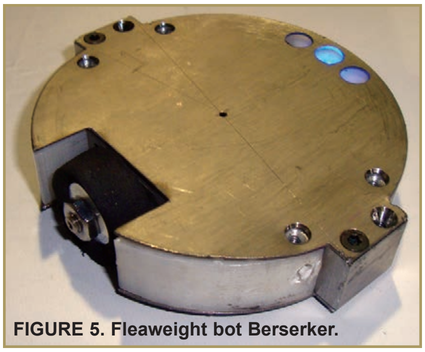


FIGURE 5. Fleaweight bot Berserker.

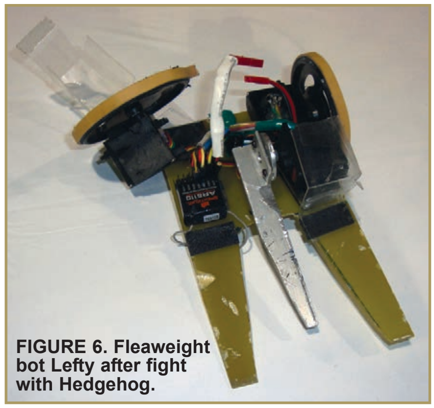


FIGURE 6. Fleaweight bot Lefty after fight with Hedgehog.

pounded each other until only one survived. It was an incredible collection of bots and a great competition.

The opening match pitted my horizontal spinner "Hedgehog" (**Figure 4**) against Team Busted Nuts' Melty Brain full body spinner "Berserker" (**Figure 5**). The match went the full three minutes, with both bots delivering and taking some awesome hits. The bots were literally flying around the arena like hockey pucks. Ultimately, the judges gave Berserker a well-deserved decision. It was the best Flea match I've ever had and a great way to start the competition. Berserker went on to the final match in the Flea competition.

Hedgehog eliminated Chris Olin's fan favorite "Lefty" — the only Flea lifter — when he cut off one of Lefty's "legs" (**Figure 6**). Chris continued to fight yelling, "It's only a flesh wound," but Lefty was finished for the day. Ultimately, Hedgehog had to tap out in a later match due to mechanical problems.

My vertical spinner "Tomahawk" (**Figure 7**) worked its way through the loser's bracket to get to the final match. Along the way, Tomahawk defeated "Invertigo" by ripping out one of its speed controllers (**Figure 8**) and "Chairman Meow" by doing pretty much the same thing to its RC receiver and battery (**Figure 9**). In the semi-final match, Tomahawk avenged a second round loss to "Dirty Sanchez" (**Figure 10**), when Dirty Sanchez got bludgeoned by one of the arena hammers and tapped out.

The championship match was another back and forth ordeal until a vicious hit from Berserker snapped the weapon motor shaft on Tomahawk sending the spinning blade flying. Although the match went the distance, Berserker was the clear winner of not only the match, but of the Flea competition.

Following the match, I asked Berserker builder Michael Gellatly to pop the top on the bot (**Figure 11**) and talk about the technology and design. The Melty Brain concept has been around for a few years and information is readily available on the Web. The concept is to spin the entire robot with one or more drive wheels so the entire bot becomes a weapon, and to use a computer and accelerometer to pulse the drive motor(s) at precise points to provide directional movement. An LED on top of the bot is pulsed to indicate the front of the bot and the forward direction of travel. The on-board computer is connected to a standard RC receiver to provide driver remote control.

Michael's bot uses a Pololu Baby Orangutan controller, an accelerometer, and open source code. I saw my first Melty Brain a few years ago at RoboGames. That bot was a larger design and drove very nicely outside the arena, but it did not do well in combat — a problem that has historically plagued the Melty Brain bots. It's obvious that Michael has cracked that nut (no team name pun intended) as his Berserker was unstoppable and clearly the best

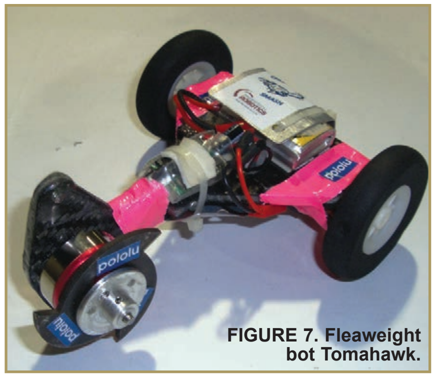


FIGURE 7. Fleaweight bot Tomahawk.

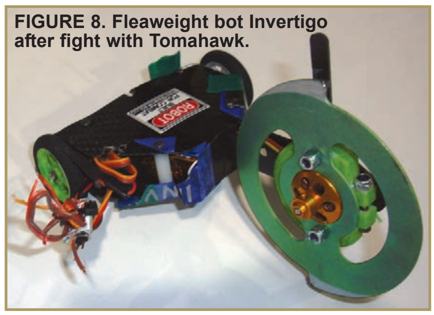


FIGURE 8. Fleaweight bot Invertigo after fight with Tomahawk.

FIGURE 9. Fleaweight bot Chairman Meow after fight with Tomahawk.

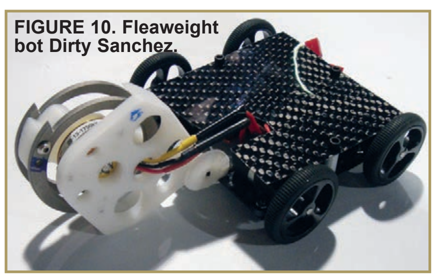


FIGURE 10. Fleaweight bot Dirty Sanchez.

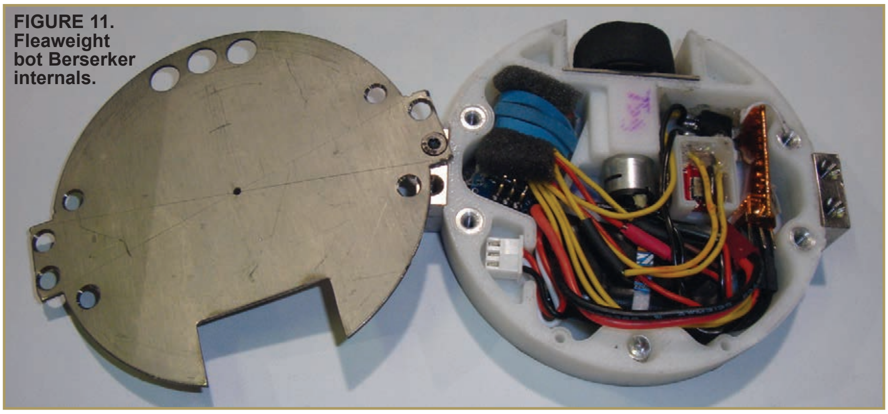


FIGURE 11. Fleaweight bot Berserker internals.

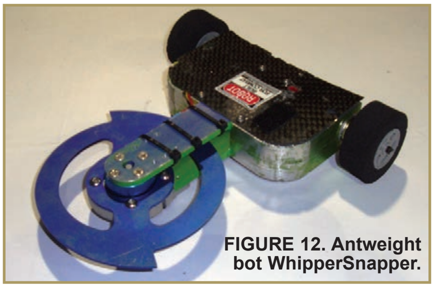


FIGURE 12. Antweight bot WhipperSnapper.

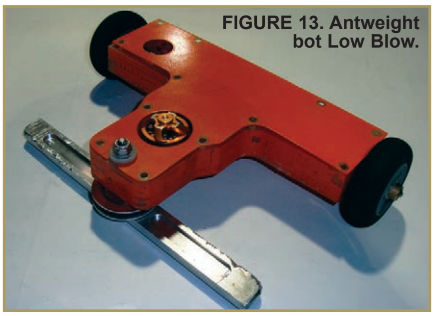


FIGURE 13. Antweight bot Low Blow.


FIGURE 14. Antweight bot Firag.

Flea in the competition.

There was one other innovative Flea weight bot — Zack Witeof's Chairman Meow (**Figure 9**). Zack also uses a Baby Orangutan controller connected to a standard RC receiver. The Baby Orangutan controller has two bi-directional motor ports that drive Pololu metal gearhead motors. The code is Pololu's 3pi RC program, which is available on their website **www.pololu.com**. The beauty of Zach's bot is the cost — Baby Orangutan controllers cost only $20. That's a cheap dual channel electronic speed controller!

The Antweight competition saw several of the 21 Ant combatants distance themselves from the rest of the field and fight it out for the gold. Andrea Suarez of Team Busted Nuts and her bot "WhipperSnapper" (**Figure 12** — shown with the blade in the horizontal position, but by design fought with the blade in both the horizontal and vertical positions) fought their way to the final match after sending high school freshman Warren Purvin's bot "Low Blow" (**Figure 13**) to the loser's bracket in what can only be described as a fluke match. After Low Blow ripped a wheel off of WhipperSnapper, the wheel got stuck under Low Blow and raised him up so his tires weren't touching the arena floor. It was a count-out for Low Blow for non-movement and another victory for Team Busted Nuts.

Fan favorite spinner "Firaga" (**Figure 14**) was eliminated early losing two bouts to wedge bots. The loser's bracket featured a lot of destruction as my horizontal spinner "Kyle's Cutter" (**Figure 15**) ripped pieces from "Ting Tang", "Captain Falcon", and "Tea Cup," forcing them all to tap out. Kyle's Cutter lost in a close decision to "Meerkat Mreow" (**Figure 16**), sending Meerkat Mreow to the semi-final match against Low Blow.

Again, Meerkat Mreow was awarded the match in another close judge's decision, setting the stage for the rematch between WhipperSnapper and Meerkat Mreow. The Ant championship match was somewhat of a letdown as less than a minute into the match WhipperSnapper gently "kissed" Meerkat Mreow with her spinning blade, and the Meerkat just quit working. I was hoping for a little more destruction since WhipperSnapper ripped the top off of Meerkat Mreow in their first match (**Figure 17**). It went in the books as a win for Team Busted Nuts and their second first place finish of the competition.

The Beetleweight competition

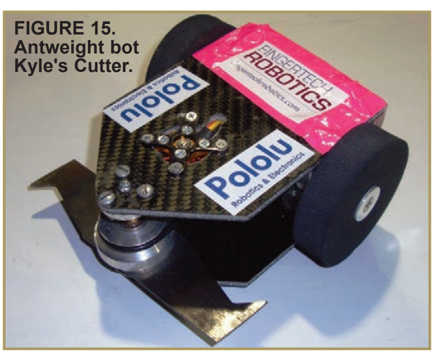



FIGURE 15. Antweight bot Kyle's Cutter.


FIGURE 16. Antweight bot Meerkat Mreow.

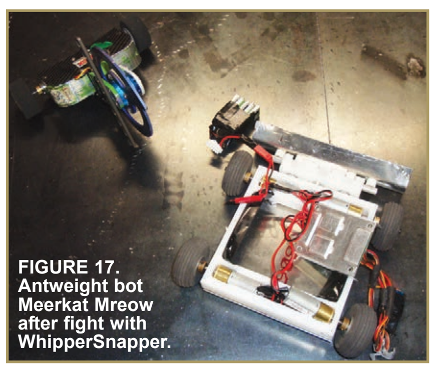


FIGURE 17. Antweight bot Meerkat Mreow after fight with WhipperSnapper.

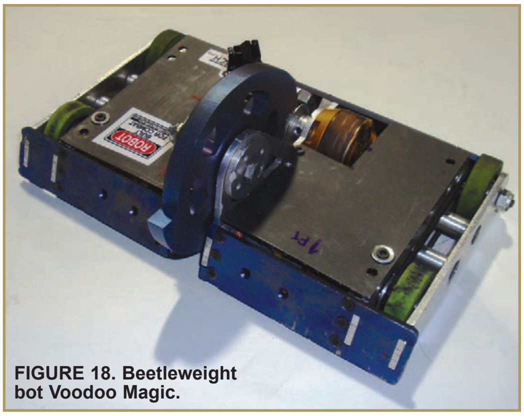


FIGURE 18. Beetleweight bot Voodoo Magic.

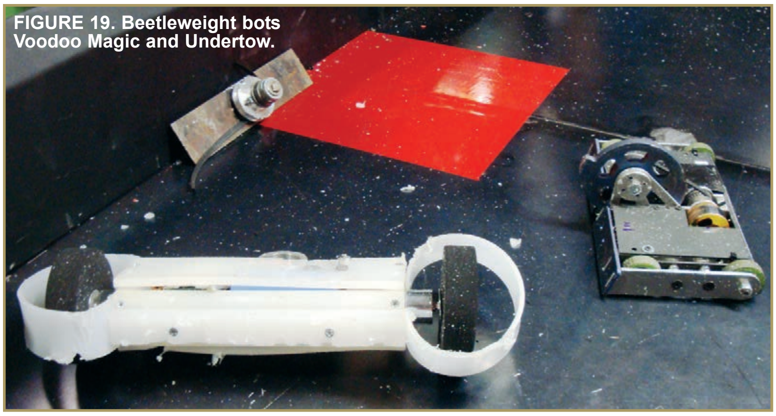


FIGURE 19. Beetleweight bots Voodoo Magic and Undertow.

was really a three bot contest between Team Busted Nuts' bot "Voodoo Magic" (**Figure 18**), Chris Olin's "Revenge of Doctor Super Brain" (hereafter referred to as the RODSB), and "Buzzkiller." Voodoo Magic started in the winner's bracket by ripping the blade off of Team Rampage Productions' bot "Undertow" (**Figure 19**), and finished the winner's bracket by manhandling the RODSB, causing Chris to totally rebuild the bot (**Figure 20**).

In the semi-final match, the RODSB eliminated third place finisher Buzzkiller to set up a rematch between Voodoo Magic and the RODSB for the Beetleweight bragging rights. This time, not only did Voodoo Magic again manhandle the RODSB, he let the arena hammers join in the fun (**Figure 21**).

Although the match went the distance, it was an easy decision for the judges. Voodoo Magic's win completed a clean sweep of first place for all three weight classes by Team Busted Nuts. It was a dominating team performance. One other Beetleweight bot merits recognition — Team Rampage Productions' innovative creation named "Torsion" (**Figure 22**).

Torsion had a spring loaded spike arm that was raised by locking two magnets — one on the lifting arm — and one on the spike arm, and then electromechanically raising the lifting arm to tighten the torsion spring until the mated arms hit a screw at the back of the bot (it's the screw with all the rubber bands wrapped around it). When the two locked arms hit the screw, it was enough force to cause the magnetically connected arms to release, resulting in the abrupt snapping of the spike arm back to its original position. While the bot performed well mechanically and was very entertaining, it failed to deliver a knockout blow and left the competition in the early rounds.

Between bouts, the event

FIGURE 20. Chris Olin rebuilds his Beetleweight bot The Revenge of Doctor Super Brain.

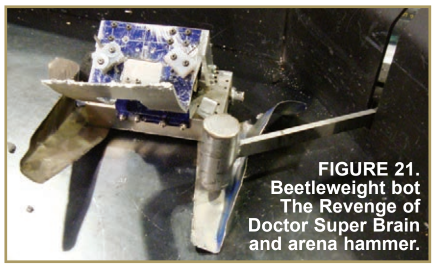


FIGURE 21. Beetleweight bot The Revenge of Doctor Super Brain and arena hammer.

FIGURE 22. Beetleweight bot Torsion.

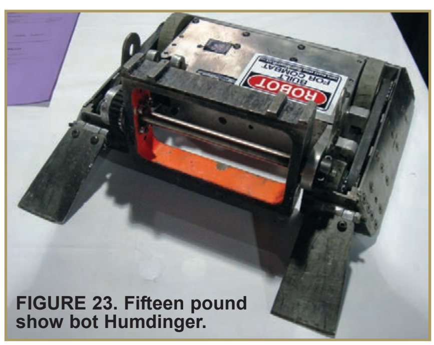


FIGURE 23. Fifteen pound show bot Humdinger.

***TABLE 1 — WINNERS AND SPECIAL AWARDS.***

| | FLEA | ANT | BEETLE |
|---|---|---|---|
| 1st: | Berserker<br>*Michael Gellatly* | WhipperSnapper<br>*Andrea Suarez* | Voodoo Magic<br>*Michael Gellatly* |
| 2nd: | Tomahawk<br>*Dave Graham* | Meerkat Mreow<br>*Zack Witeof* | The RODSB<br>*Chris Olin* |
| 3rd: | Dirty Sanchez<br>*Paul Grata* | Low Blow<br>*Warren Purvin* | Buzzkiller<br>*Curt Boirum* |

**SPECIAL AWARDS**
*Biggest Screw:* Spin Cycle
*Best in Show:* Berserker


FIGURE 24. Fifteen pound show bot Humdinger entertaining the crowd.


FIGURE 25. Event organizer Brian Schwartz, his wife Jen, and two month old babybot Zoe.

organizers called upon competitor Adam Carlson to put his 15 pound bot "Humdinger" (**Figure 23**) in the arena with old printers, monitors, TVs, scanners, and a couple of speakers to demonstrate the destructive power of a 15 pound beater bot. The crowd loved it as you can see by the expressions on their faces (**Figure 24**).

The competitors also voted on two special awards — the Biggest Screw (defined as the most damage or the toughest loss) and the Best in Show. The Biggest Screw went to a Beetle bot named "Spin Cycle," who lost two first round matches and left before I could take his picture. Best in Show went to Berserker of Team Busted Nuts. A complete list of all the winners is shown in **Table 1**.

One of the best parts of Mecha-Mayhem is that event organizers Schwartz and Miller host a free pizza dinner for the builders on Saturday night. They do it for two reasons: first, to make sure everyone gets to taste a real Chicago pizza; and more importantly, to foster the friendship and camaraderie that was felt by the original BattleBots competitors. We got a special treat at this year's dinner — Schwartz and his wife Jen introduced us to their latest creation: two month old babybot Zoe (**Figure 25**).

I'd be remiss if I didn't recognize several of the other iHobby special interest groups that had displays or conducted competitions for virtually anything with wings or wheels at the expo.

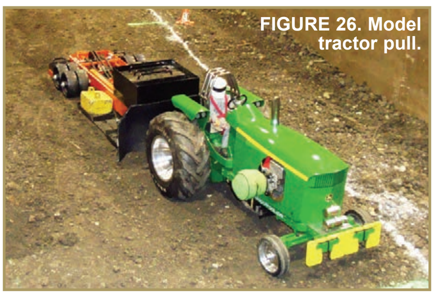


FIGURE 26. Model tractor pull.

Easily, the noisiest area was the model truck/tractor pull track (**Figure 26**). Sponsored by the National Radio Control Truck Pulling Association (NR/CTPA), this competition features RC trucks and tractors pulling weight sleds down a model track. I was impressed with the craftsmanship of these metal monsters, which usually involved the machining and mating of gasoline chain saw engines to truck/tractor drive trains (**Figure 27**). They were impressive (and loud). You can follow the NR/CTPA at **www.nrctpa.org**.

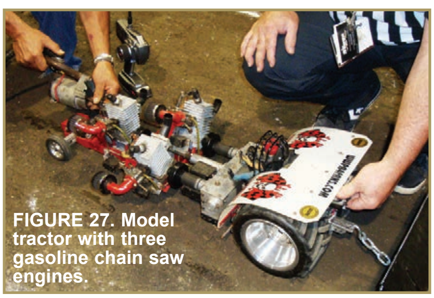


FIGURE 27. Model tractor with three gasoline chain saw engines.

I met my first "Brickologist," Beth Weis, at the expo (**Figure 28**). I asked Beth what a Brickologist was, and she confided in me that while she is a certified LEGO Professional, she just made up the name. One of the neatest things at the expo was when they turned the remote control cars loose in Beth's building area (**Figure 29**). It was the ultimate in brick destruction. You can visit Beth's website at **www.brickology.com**.


FIGURE 28. Brickologist Beth Weiss in one of her brick creations.

In addition to CRCA's Mecha-Mayhem combat robot event, the Central Illinois Robotics Club (CIRC) conducted "thinking robot" challenges, and had a number of other displays to wow the crowd including Mech Warfare (**Figure 30**), several RoboMagellan robots, advanced humanoid robots, and


FIGURE 29. RC cars destroying model brick structures.

Tim Middleton's homebrew 3D ABS printer spitting out whistles for the crowd (**Figure 31**). You can follow CIRC at **http://circ.mtco.com**.

Next year, the iHobby Expo is scheduled for October 11-14, and will move from Chicago to the I-X Center in Cleveland, OH. I'm not sure what that means to Mecha-Mayhem, but I'm hearing rumors of a collaborative fighting robot event involving Mecha-Mayhem's CRCA and Ohio's House of Robotic Destruction (HORD). Watch for it on CRCA's website at **www.thecrca.org.** SV

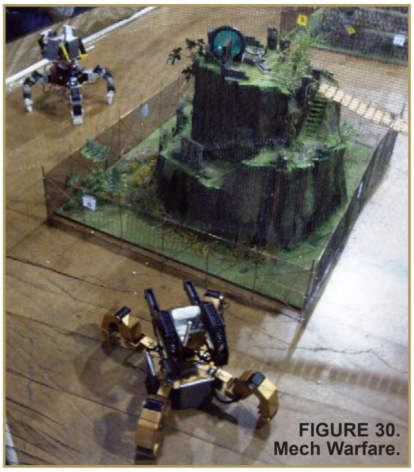


FIGURE 30. Mech Warfare.

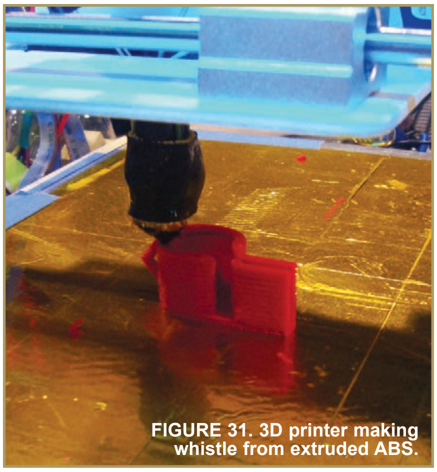


FIGURE 31. 3D printer making whistle from extruded ABS.

# The History of Robot Combat: From Humble Beginnings to Multinational Sensation

● by Morgan Berry

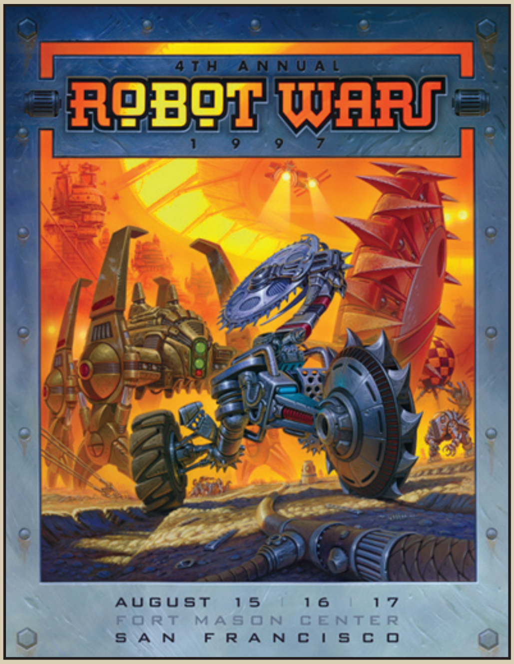
Most of the memories of my childhood are fairly typical: dance recitals, first days of school, climbing trees in the backyard, etc. Others — like a vivid memory of staring intently into a plexiglass arena while two 3 lb RC robots attacked each other — are much less common. I learned words like servo and solder, practiced tank steering through an obstacle course of orange cones in my garage, and trash talked with middle-aged nerds; I now realize none of this was a normal childhood experience.

At a young age, my brothers and I had the great fortune to become involved in the world of robot combat through my father (one of those aforementioned middle-aged nerds). The years I was involved with the sport were some of my happiest. In the haze of childhood, I had no idea that the unique niche sport I had found my way into had an equally interesting past. The small slice of the sport I had experienced was just part of a much longer history.

## The Beginning

Long before the glitz and glamour of Comedy Central's BattleBots™ brought mass appeal to the sport, robot combat had much humbler beginnings. The sport got its start in October 1989 at the ultimate breeding ground for all things nerdy: a science fiction convention. The MileHiCon — a convention running since 1969 in Denver — featured the first actual tournament promoting robotic battles called the Critter Crunch. This turbulent start is outlined in Brad Stone's book *Gearheads* — a fantastic read for anyone interested in the history of robot combat.

**Participants at the 1989 Critter Crunch at MileHiCon in Denver.** ***Photo courtesy of Wired.com.***




*Courtesy of MarcThorpe.com; Copyright Marc Gabbana.*

According to Stone, a small group of mechanical engineers known as the Denver Mad Scientist Club envisioned the event after viewing videos from the robotic performance art group known as the Survival Research Laboratory and learning of a competition at MIT that required homemade robots to compete in mechanical tasks like collecting ping pong balls. By coupling these influences with the already existing Critter Crawl at the MileHiCon — which a recent *Wired* article about the event describes as a "sort of beauty pageant for windup toys and remote-control gizmos" — the group created a robot combat event like no other.

It was part eccentric spectacle, part brutal fight to the death, and all fun. This formula has carried through the sport to this day; I personally recall a bot in my robot combat league named "Cousin It" that used a cheap plastic hat as its armor which would invariably be ripped to shreds by the end of the competitions. Unlike the behemoth bots featured on BattleBots, these bots could be no bigger than one cubic foot; although, once the match started they could be expanded with appendages.

The bots could weigh no more than 20 pounds. The matches were fought on a folding table, and with spectators only a few feet away from the bots, the weapons had to be kept relatively tame. That did not, however, stop some weapons like a flamethrower or pneumatic ram from occasionally sneaking in.

Since its start in 1989, the Critter Crunch is an ongoing event at the MileHiCon that continues to appeal to eccentric, destruction-loving, creative types like those who first envisioned it.

## Meanwhile ...

Eventually, word of the Critter Crunch began to spread and other conventions around the country began hosting robot combat events as well, most notably at DragonCon in Atlanta. While these non-commercial events thrived in the convention world, another man was independently developing a bigger picture view of robot combat.

1995 Robot Wars San Francisco. *Courtesy of MarcThorpe.com.*

Marc Thorpe was a San Francisco based animatronic designer who created special effects for the last two movies in the Star Wars trilogy for LucasFilm, which in itself is enough to inspire awe in even the most casual nerd. According to his own website **MarcThorpe.com** — while independently working on a remote control vacuum cleaner in 1992 — Thorpe looked at the device and realized its sinister potential to be transformed into a metal-crunching death machine. Using this idea and the entrepreneurial spirit he had gained from working at LucasToys, Thorpe realized the money-making potential behind a remote-controlled robot combat tournament and immediately began this new business venture.

After a few unsuccessful attempts to get the event going — which he had named Robot Wars — a feature in a 1994 issue of *Wired Magazine* finally gave Thorpe the monetary support he needed to put on the competition. This event was markedly different from the Critter Crunch.

For one thing, it was independent and was not confined to the hotel ballrooms of science fiction conventions; as a result, the robots could be much larger and more dangerous. The Robot Wars brand was continually apparent through specialized posters, t-shirts, and trophies. Some notable competitors at the first event were Caleb Cheung, inventor of the wildly popular Furby toys, and Will Wright, the creator of the Sims video games. Following the massive success of the first competition, the events continued until 1997; each larger and more exciting than the last.

## Robot Wars Expands onto Television

The business partnership between Thorpe and Profile Records began to sour as Profile Records pursued the creation of a Robot Wars television show in the U.K. Despite legal battles surrounding ownership of the brand, the series

was aired in 1998 on British television. This version of Robot Wars was much different from the San Francisco competition, and completely unrecognizable from the simplicity of the Critter Crunch.

The original season involved three competitions for the robot builders. The first — called "The Gauntlet" — was an obstacle course. The second — "The Trial" — featured games like "Football" and "Tug of War." The last challenge was "The Arena," which required the bots to compete in the familiar combat based battles. While the arena for the Robot Wars competition in San Francisco had originated with just a two foot high plywood barrier between the audience and the bots, the television version had a complex arena with various dramatically named hazards to add another layer of difficulty to the competition. The Pit of Oblivion, the Disc of Doom, and the Floor Flipper were just a few of the hazards the bots would face.

Another notable difference from previous competitions was the addition of "House Bots" which were not confined to the weight or weapon constrictions of the competitors, and could attack the competing bots if they ventured into certain zones of the arena. The show proved to be a wild success, spanning seven seasons and sparking multiple spin-off shows and a line of "House Bot" children's toys.

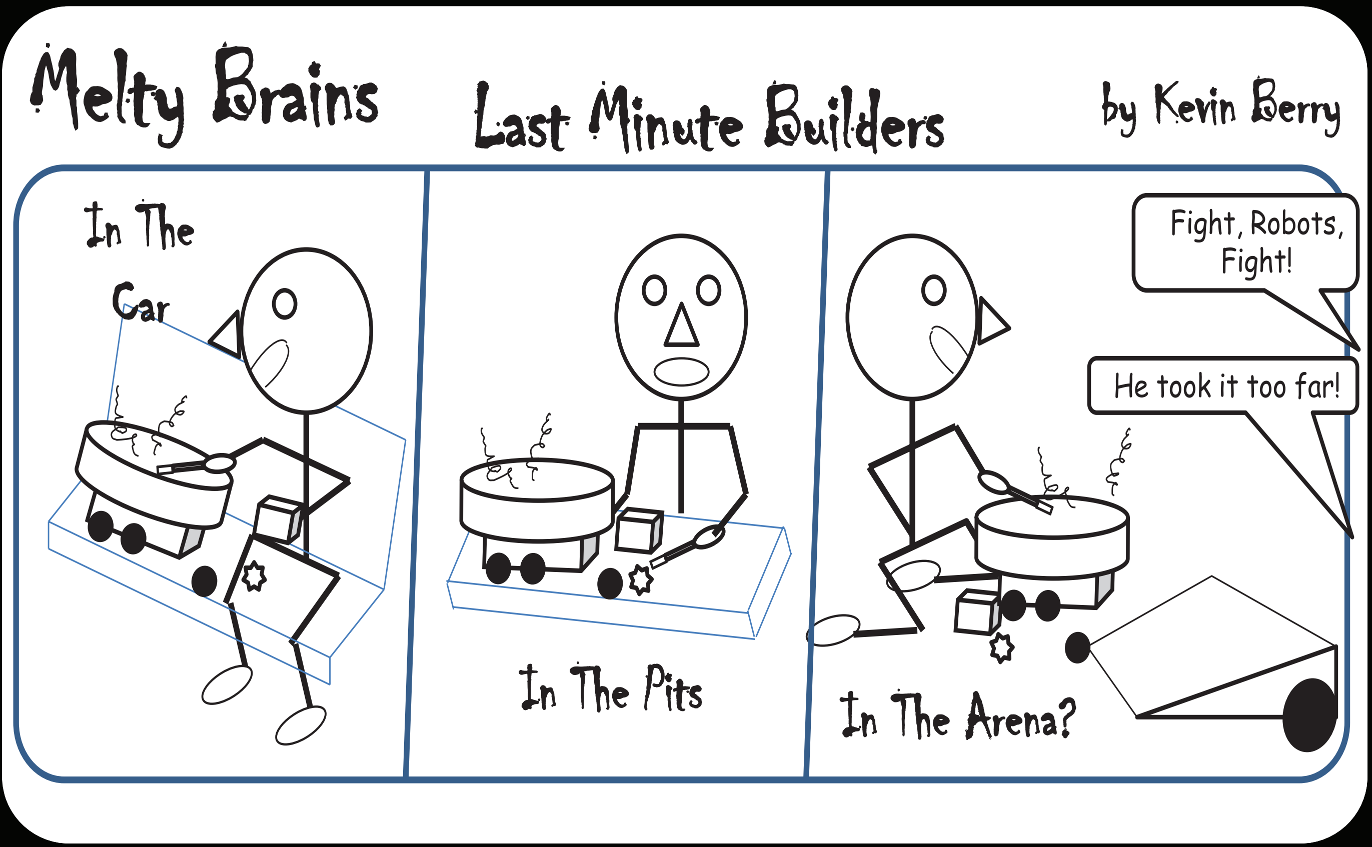
## Legal Turbulence Sparks Greatness

As an event timeline from **RobotCombat.com** outlines, as the British television show began to takeoff, legal battles hindered the growth of robot combat in the United States. Back in San Francisco, those involved in the original Robot Wars competition were preparing for Robot Wars '98 when Profile Records issued a court order barring Thorpe from holding the event. Profile also attempted to shut down the Robot Wars Internet forum the group had created.

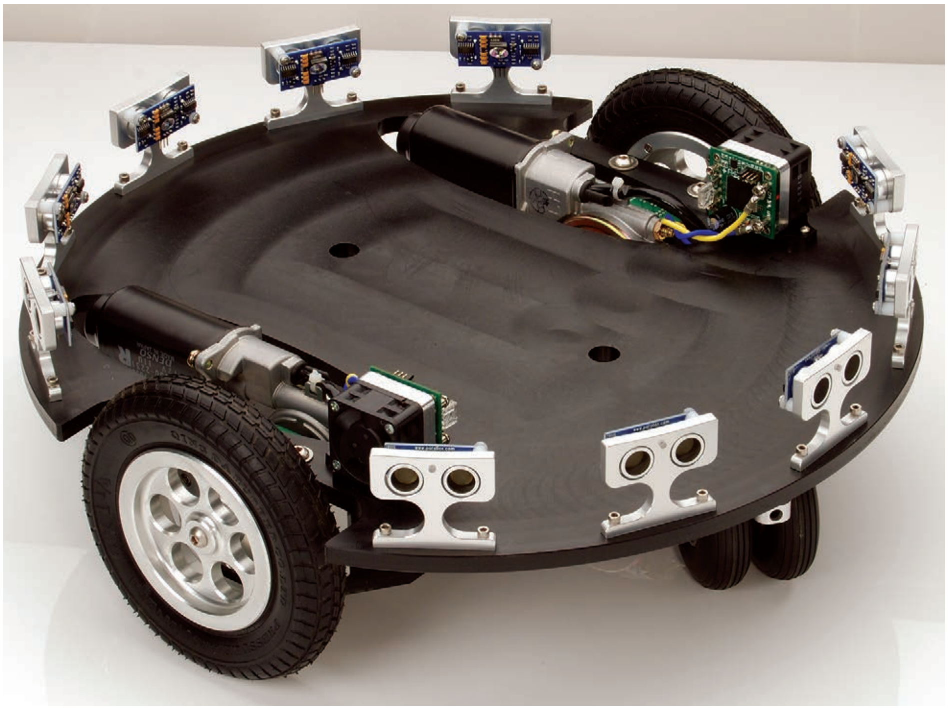
To prevent the builder's hard work from going to waste, an invitation only, spectator-less event called Robotica was organized, only to once again be barred from taking place by Profile. Eventually, Thorpe would also lose his ownership of Robot Wars, and when veteran robot builders Trey Roski and Greg Munson attempted to organize an entirely new competition — the now famous BattleBots — they were also sued by Profile Records.

The future of robot combat was beginning to look bleak when, finally, the sport's luck began to change; the court had ruled in favor of BattleBots. With a break in the legal battles finally allowing competitions in San Francisco again, the golden age of robot combat was about to begin.

Coming up in the next installment of this series will be The History of Robot Combat: BattleBots Brings Mass Appeal to the Sport. SV



# Parallax MadeUSA — The Big Robot That Could



by Gordon McComb

www.servomagazine.com/index.php?/magazine/article/january2012_McComb

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

FIGURE 1. The completed MadeUSA, with base, motors, motor drivers, and ring of 10 Ping))) ultrasonic sensors (mounted on stands, also included).

**Sure, I get to write about robots. However, that doesn't mean I always get the chance to play with them. And by play, I mean truly having fun — the kind of fun a kid has when given a new toy for Christmas.**

Though definitely fun, my latest robot — the MadeUSA from Parallax — is hardly a toy. It measures 18" in diameter and weighs over 30 pounds when fully decked out. The bot is provided as a do-it-yourself base (AYOM: Add Your Own Microcontroller). I've tried it with an Arduino and a Parallax Propeller, and both were easy to integrate.

The name MadeUSA comes from a play on words, combining "Made in USA" with Medusa — the horrible looking creature from Greek mythology known for having snakes for hair. Take one look at her and you'd be turned to stone.

I haven't noticed any rock-forming tendencies while working on my MadeUSA, but I've nicknamed my robot "Big Brute," and here's why: Though Parallax doesn't mention it, this bot is strong enough to cart around a full grown person. If nothing else, you could use MadeUSA to build a robotic golf cart for your Great Dane. Yeah, that's one big dog!

## It All Started With a Piece of Wood

The original MadeUSA robots were made from high grade plywood. Now, Parallax manufactures their big bot out of a 5/8" thick slab of machined high density polyethylene (HDPE). A completed MadeUSA kit — with all included trimmings — is shown in **Figure 1**.

A computerized cutting mill shapes and forms the main deck of the MadeUSA. Though a very strong plastic, HDPE is known for being easy to drill. That makes it simple to customize your MadeUSA with your own accessories. The base is cut into a circle, with openings to make room for the two high power motors (see **Figure 2**). The motors are new automotive power window actuators, and have got plenty of torque.

Smaller holes are pre-drilled for a ring of 10 Ping))) ultrasonic sensors. These sensors attach on custom stands, and are placed evenly around the circumference of the MadeUSA base. I imagine that it's the Pings))) around the

FIGURE 2. The MadeUSA base is constructed from 5/8" HDPE. A separate battery compartment deck is laser-cut from sturdy cast acrylic.

base that give the MadeUSA its *Medusa* look. I elected to not use all 10 for Big Brute, saving a few for other projects.

(If you don't want or need all 10 Pings))), you can purchase the MadeUSA component parts separately which shaves several hundred dollars off the total price. Check out the MadeUSA page on the **Parallax.com** site for a listing of the separate components. Mix and match what you need.)

Air-filled pneumatic tires are provided with the kit, along with custom machined aluminum wheel hubs. To prevent excess stress on the motor shaft, the MadeUSA uses a heavy-duty pillow block bearing for each wheel. (The block is sturdy enough to support a horse, let alone an 18" robot.) All the weight of the robot bears down on this bearing, literally. That helps to prolong the life of the motors, and keeps the motor shafts parallel with the base.

## Getting From Here to There

Power for the motors comes from a 12 volt seven amp-hour battery that you provide. The MadeUSA kit comes with a clever battery holder that adjusts for variations in size and shape of the battery, holding it snugly between a set of aluminum standoffs. (See **Figure 3** for details.) The battery holder is placed on the underside of the robot which keeps its center of gravity low. That makes it highly unlikely the MadeUSA will tip over, even on a steep incline.

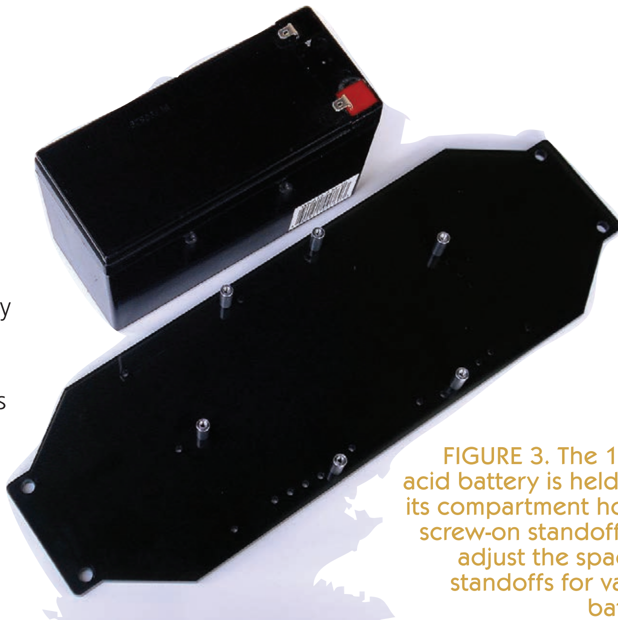


FIGURE 3. The 12 volt lead acid battery is held captive in its compartment holder using screw-on standoffs. You can adjust the spacing of the standoffs for variations in battery sizes.

Complementing the heavy-duty motors are a pair of 25 amp motor drivers, Parallax model HB-25. By themselves, these drivers are operated as if they are radio control (R/C) servo motors. Provide a train of pulses of 1,500 microseconds and the motor stops. Shorter pulses make the motor turn one direction; longer pulses make it turn in the other direction. Speed control is affected by altering the pulse width.

Should you be interested in such things, the fan cooled HB-25 uses mini-blade style fuses. The fuse will blow should there be a short or other serious condition in the motor or its wiring. Each HB-25 comes with 25 amp fuses pre-installed.

The HB-25 drivers are augmented by a set of precision wheel encoders. The motor and position encoder kit is shown in **Figure 4**. These encoders are commanded through standard serial signals. You can opt not to use the encoders — you can always control the HB-25s directly. Since the encoders provide a means to accurately maneuver the robot, there's little reason to disregard them. In use, your microcontroller is coupled to the wheel encoders, and the wheel encoders are connected to the HB-25 drivers. The HB-25 drivers, in turn, are attached to the motors.

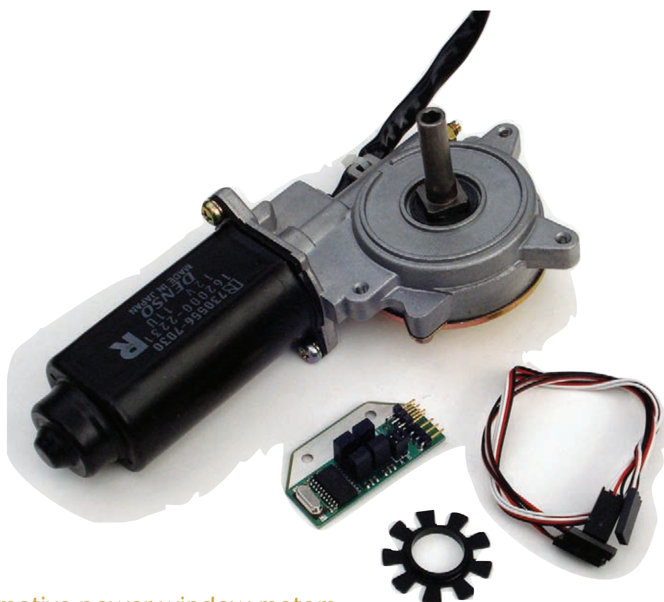


FIGURE 4. MadeUSA uses two automotive power window motors for locomotion. Precise control is provided by an optical encoder.

FIGURE 5. The MadeUSA robot uses custom-made pneumatic wheels. You can adjust the air pressure for a firm or soft "ride."

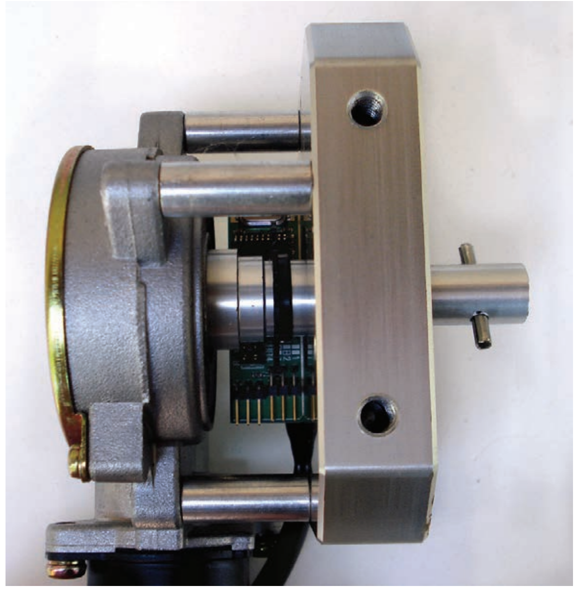


FIGURE 7. Carefully align the encoder disc so that it's between the sides of the slotted optical switch. Gently push the disc back and forth along the wheel shaft.

The wheel encoders on the MadeUSA operate a bit differently than what you might find on most other robots. The typical encoder arrangement provides velocity control; in typical use, the microcontroller on your robot monitors the timing of the encoders from both wheels, and makes on-the-fly speed corrections to keep both wheels turning at the same rate.

On the MadeUSA, the wheel encoders are engineered to provide position control. Rather than simply tell the robot to go forward, you specify *how far* to go forward. The encoders keep track of the distance of both wheels. When the robot has reached its destination, the motors are *automagically* turned off.

The wheel encoders also provide a means to control not only the speed of travel, but also acceleration and deceleration. Suddenly starting and stopping a robot as large and heavy as MadeUSA can put a strain on the motors and other mechanics. In operation, the wheel encoders command the HB-25 drivers to ramp up to speed for regular travel, then slow back down again as the desired position is reached.

You send simple serial commands to the wheel encoders to control the various aspects of travel. That should suffice for most applications, but if you want to go further, you can reprogram the microcontroller on the wheel encoder board so that it behaves in whatever fashion you wish. The design of the encoder and its firmware are open source. You can download full schematics, layout, and source code from **Parallax.com**.

## Building the MadeUSA Kit

MadeUSA comes as a kit, and building it requires a solid couple of hours. Before starting, review the assembly documentation that comes with MadeUSA to determine if you have the proper tools. At a minimum, you'll need a basic assortment of standard hex wrenches, and a #1 Phillips screwdriver.

After familiarizing yourself with all the components, assemble the wheels first — they have the most parts. The wheels (**Figure 5**) are constructed with a pneumatic inner tube, rubber tire, custom machined aluminum hub, aluminum rim, and an axle. When

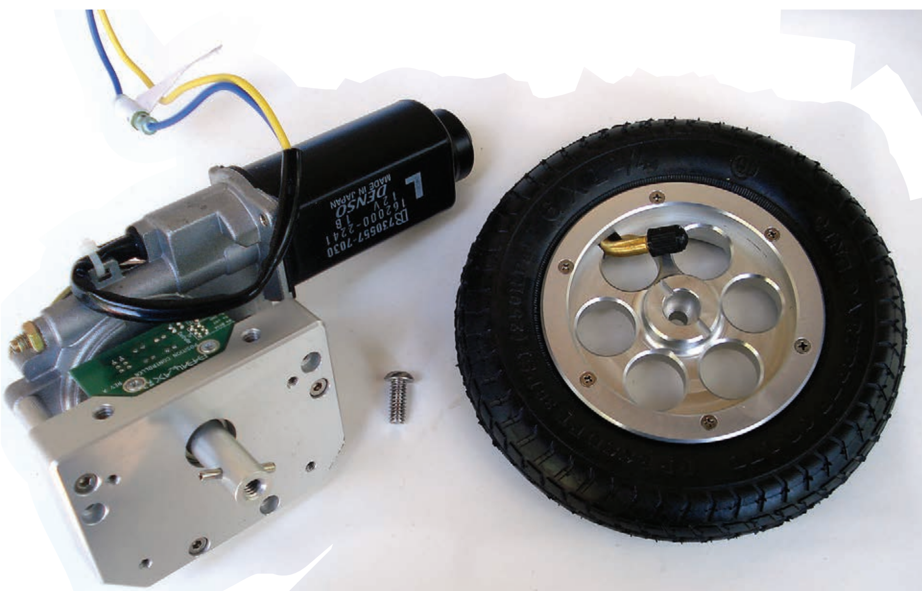


FIGURE 6. The completed motor assembly, showing the heavy-duty pillow block and assembled wheel.

FIGURE 8. One of two casters on the MadeUSA. These swivel freely and give the robot support in both the front and back.

assembling the inner tube onto the hub and rim, be sure none of the rubber is pinched or else it could cause a leak when you add air to the tire.

And, speaking of adding air ... best use a hand pump. It's hard to control the amount of air that goes into the tire when using a powered air compressor. Even a short blast of air might over-pressurize the tire.

The motors and pillow blocks can be constructed next. The completed drive assembly with pneumatic wheel is shown in **Figure 6**. Before attaching the motors to the base, however, read up on the jumper settings for the position encoders. A jumper is provided to select one of several Device ID addresses, should you wish to daisy-chain the two encoders off one serial port. They're shipped from the factory to both use Device ID 1; if you'd like to use one serial port for both encoders, set one of them to Device ID 2.

Caution: When mounting the plastic codewheel onto the wheel shaft, be sure to apply *equal pressure* to just the hub — don't press on the outer flange of the codewheel. You don't want to break off any of the "vanes" of the codewheel. Once seated on the shaft, turn the assembly sideways and sight down the center of the optical switches. Ensure that the codewheel rotates within the middle of the two infrared slot switches (see **Figure 7**). You'll hear a scraping or tapping sound if the codewheel is rubbing against anything.

I found it easier to install all the wiring for the wheel encoders and motors at this point, before mounting the motors to the robot. Once the wheels are mounted, the headers on the wheel encoders can be hard to reach.

The MadeUSA uses two double-wheel casters for front and back balance. These casters are constructed from commercial rubber wheels (see **Figure 8**) and custom machined parts to create a very sturdy and well made mechanism. The caster swivels on a center axle; the whole thing is kept in place with a collar and setscrew. The caster parts are shown in **Figure 9**.

The only real trouble I had in assembling my MadeUSA was getting the setscrew for the axle in place. Mine used a slotted setscrew (as opposed to a hex socket setscrew), and the screwdriver wouldn't keep the screw captive long enough to thread it. In the end, I solved the problem by putting a bit of 3M tacky putty over the head of the screw. That kept the screw on the screwdriver long enough to thread it into the retainer.

## Completing the MadeUSA

The MadeUSA is a starter base; it isn't a complete robot, ready to roll right out of the box. What's not included: At a minimum, you need a microcontroller, batteries, power wiring, and switches. Your choice of microcontroller and other electronics is up to you. So far, I've built the MadeUSA in two versions; the first (shown in **Figure 10**) used an Arduino Mega 2560, along with a combination MP3/MIDI shield from SparkFun.

My current build uses a Parallax Board of Education (PropBOE) which has the same footprint as the BASIC Stamp BOE, used on the Parallax BOE-Bot. Instead of a BASIC Stamp, the PropBOE uses a Propeller microcontroller. One of my favorite features of the PropBOE is that it comes with a socket for an XBee radio transceiver. Just plug in the XBee, and you're ready to talk to other devices — no additional wiring necessary. The PropBOE also has connections for six R/C servos and has its own micro-SD card reader, audio amplifier, video generator, and analog-to-digital converter.

Both versions I've built use a separate deck constructed with 6 mm expanded PVC plastic; the basic idea is shown in **Figure 11**. This allows me to easily swap electronics, without having to do a major mechanical overhaul. On the main deck is the microcontroller plus solderless breadboard area. It's for experimenting before I commit any electronics to a permanent circuit board.

FIGURE 9. Caster parts before assembly. After some cussing (and some sticky putty), I finally conquered the small setscrew shown in the foreground!

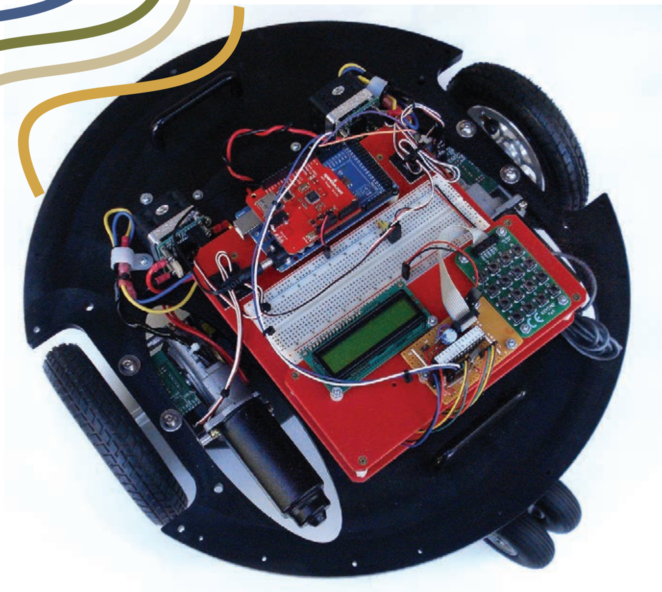


FIGURE 10. One of the versions of my MadeUSA which I call Big Brute. This one uses an Arduino Mega 2560 microcontroller. The solderless breadboard is for experimenting. The LCD and keypad are controlled by a chip-only Arduino, connected serially to the Mega 2560.

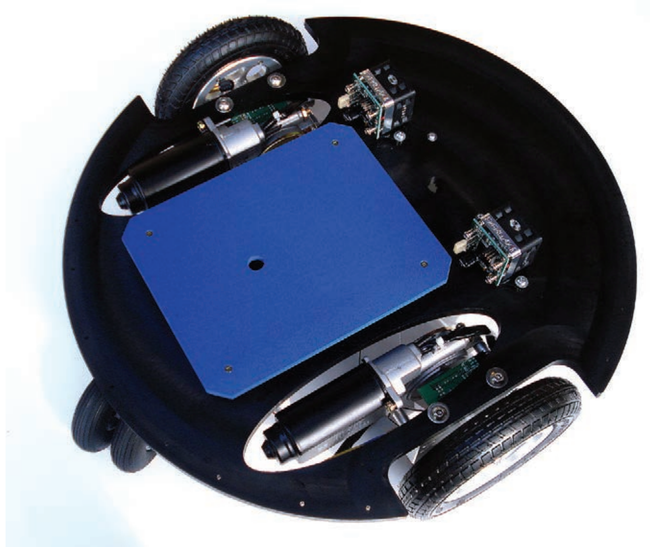


FIGURE 11. MadeUSA platform with a smaller deck mounted on top, to make it easier to add (and remove) electronics and other components. The deck is attached using standoffs.

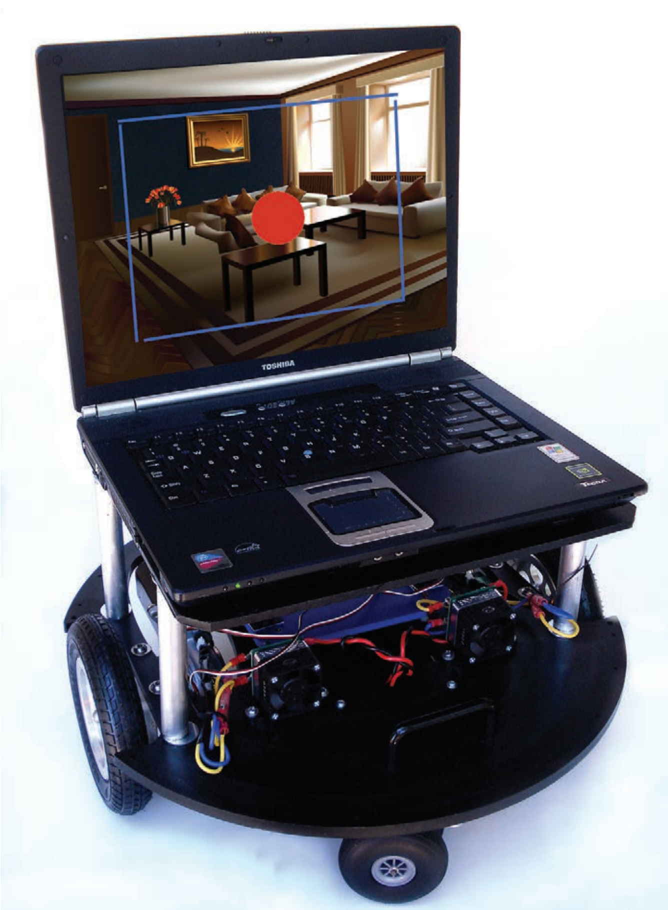


FIGURE 12. My Big Brute, with separate laptop deck and laptop. (Yes, the picture is simulated. This is before giving the robot any kind of vision system which is fairly easy to do using a webcam and vision software, such as RoboRealm.)

The decking approach also helps increase the surface area which maximizes mounting space. Under the PVC deck is a Parallax lithium-polymer battery pack, combining both the batteries and recharging circuit on one board. There's also an amplifier and speaker with volume control. It's for the MP3 and MIDI sound effects.

Additional sub-decks are provided for still more electronics. An LCD panel and keypad provide for user interaction. I made the interface to the panel and keypad using a chip-only Arduino. I programmed an Arduino board that used the DIP version of the Atmel ATmega328P chip. Once programmed, I pulled the chip off the board and installed it in my own circuit. (You can get replacement '328P chips with the Arduino bootloader already programmed in, from SparkFun and Adafruit.)

Finally, because its drive system is capable of hauling around considerable weight, the MadeUSA is an ideal robot for connecting to a laptop as a master control. Just add another deck on top, using heavy-duty columns or standoffs. **Figure 12** shows such an arrangement with 5/16" carriage bolts for the columns. I made the top deck from 1/4" HDPE. Also notice I put small handles on the top of the base, to assist in carrying it around.

## Connecting With the Kinect

The MadeUSA is a flexible base for hobby, education, and research. It is priced hundreds — if not thousands — of dollars less than similarly equipped research bots, making it ideal for cost-conscious experimenters, teachers, and students. The MadeUSA concept caught the attention of Microsoft, who worked with Parallax to develop a special version of the robot for use in the latest incarnation of Microsoft's Robotics Developer Studio (RDS).

That version — known as Eddie — comes with an

integrated central processor that also includes motor drivers. The Eddie kit — fully assembled with extra-cost laptop and Kinect — is shown in **Figure 13**.

Eddie is intended to be used with a laptop running Windows 7 and RDS, making it compatible with the Microsoft Kinect 3D motion sensing device. Services running in RDS allow you to use a Kinect and your robot for mapping, motion planning, obstacle avoidance, even human control interfacing.

*(Take note: You need Windows 7 if you plan on using a Kinect. If the Kinect isn't critical for your application, you can fall back to an earlier version of RDS which can run on Windows Vista or even XP.)*

Eddie also varies from the MadeUSA in that a second round deck is included, along with support columns for both the second deck and the Kinect hardware. Rather than using 10 Ping))) ultrasonic sensors, Eddie is happy with just two Pings))) plus three Sharp infrared proximity detectors. The ultrasonic and IR sensors are designed to augment Kinect's vision, as they're more adept at spotting small low-lying obstacles directly in front of the robot.

At $849 for the MadeUSA and $1,249 for Eddie, these robots are for the more serious tinkerer. Thanks to their hefty construction and ample mounting space, these are bots you can rely on for many years. Future plans for my MadeUSA include a five degree-of-freedom arm on the back, a homing beacon for automatic recharging, and an optical line follower — football field size!

FIGURE 13. The Parallax Eddie platform, designed to complement the Microsoft Robotics Developer Studio platform. The laptop and Kinect are extra-cost options, but the Eddie kit includes most everything else, including a special central controller.

Some day I'll even build that robotic golf cart for my dog. **SV**

Gordon may be reached at rbb@robotoid.com.

# Getting a Grip on Jamming

by Charles Ford

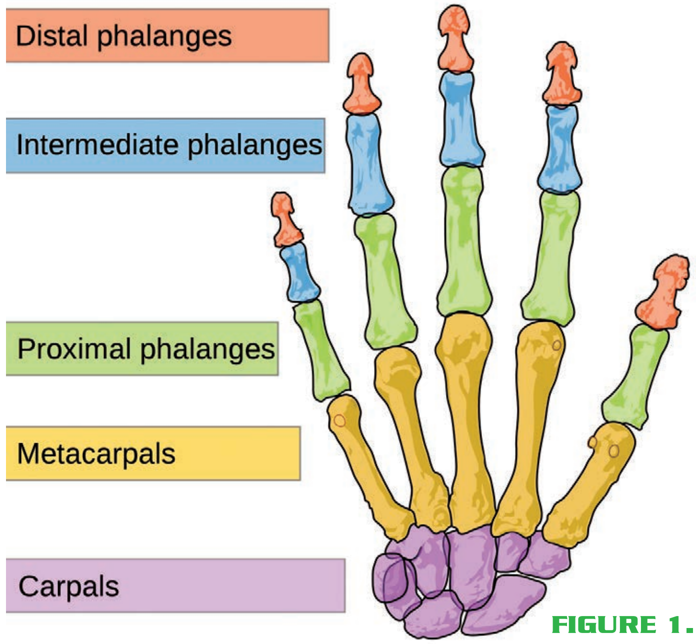



FIGURE 1.

You are probably well aware that daily, trivial tasks for us humans present major headaches for robots. If you are at a desk, close your eyes and reach out and grab a pen or pencil. When your fingers brush up against it, you take your mental image of the object, its shape, orientation, smoothness, mass, etc., and use that information to grip it. You can easily decide how much pressure to apply and how to orient your fingers for the best grip. For a robotic hand, this is a major challenge.

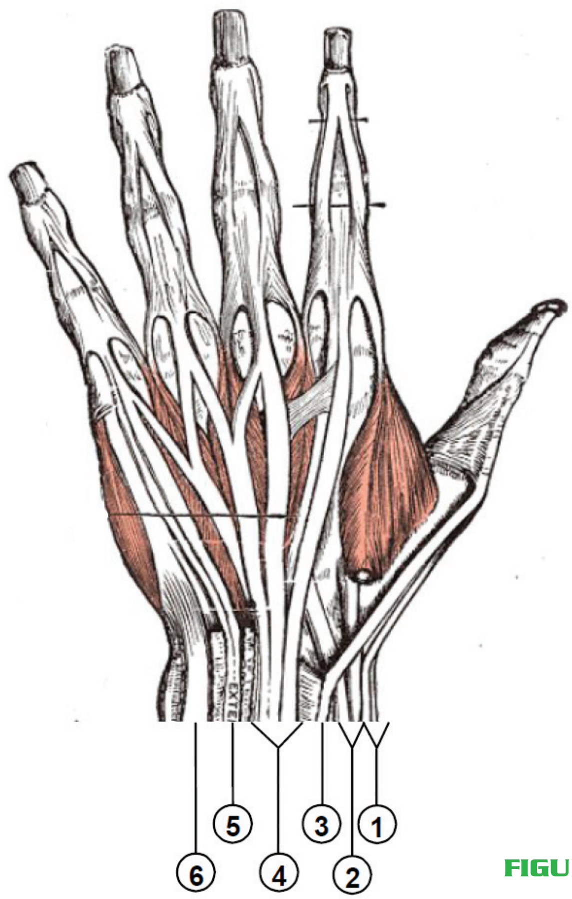



FIGURE 2.

FIGURE 3.

FIGURE 4.

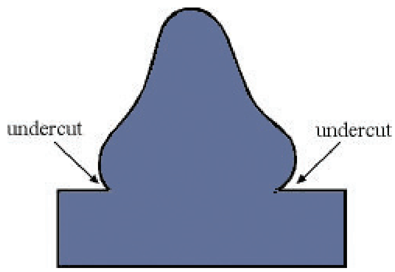
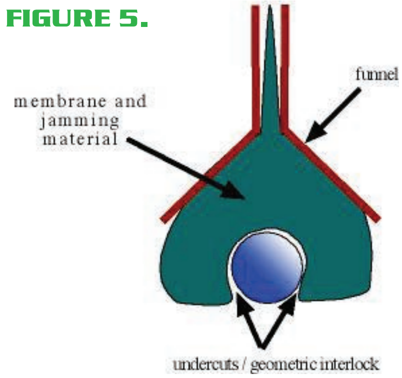



FIGURE 5.




The human hand has 27 bones: 19 in the hand itself and eight in the wrist [**Figure 1**]. To motivate these bones, you have about 40 tendons, along with 20 muscles in the hand and 20 in the wrist [**Figure 2**]. Just think of the number of degrees of freedom! When you cup your hand around an object the orientation of your fingers changes in subtle and extraordinary ways. Now, consider the robot hand [**Figure 3**]. No contest!

Researchers at the University of Chicago, Cornell University, and iRobot teamed up to research ways that robots could pick up small everyday objects reliably. If robots are to one day be helpmates to the elderly and the disabled, they will need to be able to pick up medications and other small, irregularly shaped objects.

## Get Your Jam On

Jamming describes the process that makes your vacuum-packed bag of coffee seem solid as a rock until you break the seal. Loose granular material resting in an elastic bag is at the threshold between a flowing state and a rigid state. If you create a vacuum in the elastic bag, the granular material becomes rigid. Normalize the air pressure, and you have unjammed material that can flow once again.

What creates strength in the jamming process is the target object's undercuts. Undercuts are the bane of mold makers and **Figure 4** shows how molding a nose prosthetic has problems with undercuts. If you are making a cement negative mold of a life cast, you have to deal with many locations on the face where undercuts occur. If you ignore the fact that the nostrils curve back inward, you will create a negative mold that locks onto the life cast and cannot be removed without damaging the molds.

Undercutting is called "geometric interlocking" in the engineering world. Undercuts are exactly what the jamming process needs. When you apply pressure to the elastic bag filled with granular material, the bag membrane will stretch around, and the unjammed granular material will flow around the object. Once you create a vacuum in the bag, the granular material becomes rigid, and if the object is undercut enough, you will have a very firm grip on it [**Figure 5**].

A second mechanism in the jamming process is friction. There will be a certain amount of friction between the object and the gripper's skin material. This friction will also help the gripper keep a firm grasp.

If the gripper can flow around an object enough, you will get a suction effect. This third mechanism also helps keep the object firmly held. Basically, the universal gripper is a passive device. Once the suction is applied, it becomes a rigid mass with no moving parts. The three mechanisms of geometric interlocking (taking advantage of undercuts), friction, and suction combine to create a more powerful hold than you would expect.

My journey into this realm began with a brief news article. The article referenced a YouTube video (**www.youtube.com/watch?v=0d4f8fEysf8**) and after watching it a couple of times, I was hooked. (There is also a longer article with pictures on the Singularity Hub website at **http://singularityhub.com/2010/10/27/irobots-universal-gripper-pours-a-drink-and-draws-shapes-video**.) It looked so simple that I couldn't believe it had taken so long for someone to implement it.

FIGURE 6.

FIGURE 7.

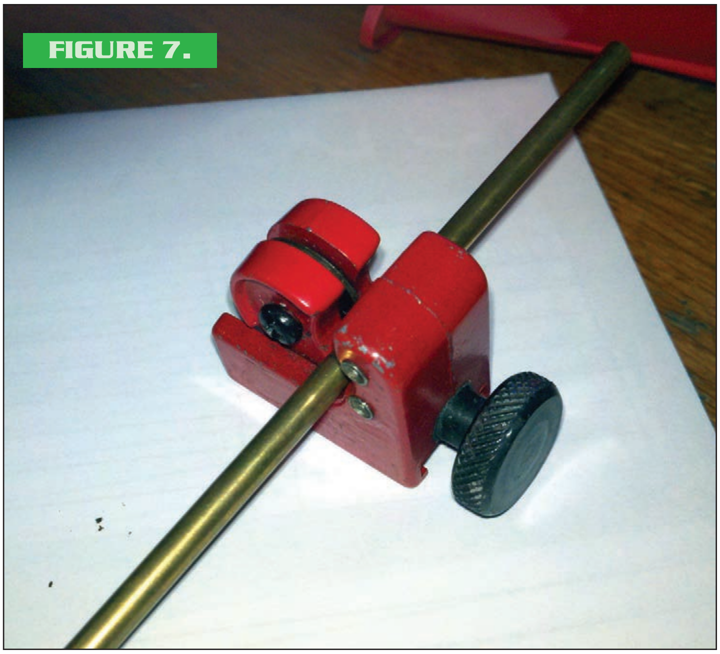
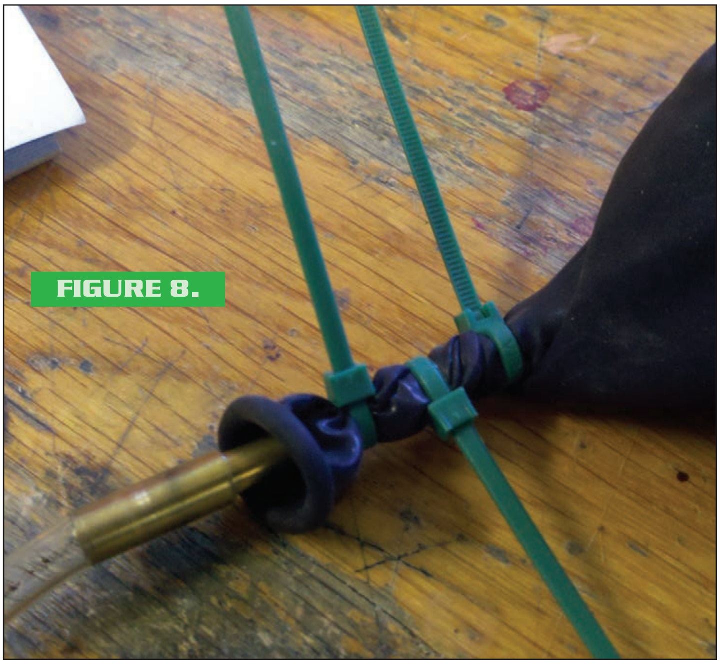


FIGURE 8.



## Get a Grip

I gathered the materials listed in the **Sidebar** and decided to create a gripper of my own that could be incorporated into one of my robot designs.

Adding a vacuum pump or a powerful air mover seemed difficult and probably unnecessary. Any sort of suction pump would add weight to a robot, and use up precious electrical power. In addition, it seemed wasteful to have a pump running the whole time an object was grasped. I thought of using solenoid-powered valves, but the suction pump just seemed too wasteful.

So, I started to think about vacuums.

## Suck It Up

I remembered that when they were filming "An American Werewolf in London" that Rick Baker used hypodermic syringes hooked up in a line. When a really big syringe was compressed at the end of the line, the little ones popped up creating the appearance of a spinal column erupting below the skin of the creature's back. The tubing connecting the syringes created a closed system that seemed to be pretty elegant.

I decided that the same idea might work with my universal gripper.

In addition, I thought that using a funnel could work nicely for balloon support. Working through some thought experiments, I saw an issue with the initial contact between the gripper and the goal object. In the video, a robotic arm moved the gripper into position and brought it down. Each time, the goal object was on a flat surfaced tabletop. So, what if you were dealing with objects at different heights or on uneven surfaces?

I realized that adding a pressure sensor inside of the funnel would allow the Arduino to apply suction only when the balloon came into contact with the object to pick up. This led me to Plusea and her instructables article (**www.instructables.com/id/Conductive-Thread-Pressure-Sensor**) using Velostat by 3M, conductive thread, and neoprene. It seemed that I had all of the solutions before me.

FIGURE 9A.

FIGURE 9B.

I went to a store that inflates balloons with helium. I reasoned that their balloons would be newer (latex ages and rots) and more flexible than any out of a pre-packaged bag. I was worried that the latex balloon might have small pin-holes that would affect the rig's ability to create a good vacuum. I decided to make a test. I inflated the two latex balloons I purchased and put a clothespin on the bottom. The next morning I found that one balloon had deflated by half. The second balloon was still fully inflated, so I used it [**Figure 6**].

I filled the balloon with coffee. I chose the cheapest and coarsest coffee in the store. I think that a coarser grind would provide more jagged edges for the coffee granules to lock onto each other when the vacuum is applied. I didn't test it, so it remains a theory. After reading an article in *The Proceedings of the National Academy of Sciences* (see **Sources**), I found that some researchers have used tiny balls as a jamming medium. What seems to be important is that the medium can flow under normal air pressure.

I decided to only fill the balloon about 3/4 full of coffee. When the vacuum is removed, the coffee needs some room to become a flowing material again. I also reasoned that there would need to be some room within the membrane for the coffee to be displaced as the gripper enveloped the object to be captured.

Next, I used a craft pipe cutter to remove about five centimeters of the brass pipe. You can get pipe cutters from hardware and craft stores. You could also use a Dremel tool or even a hacksaw. I wanted a rigid tube that would not deform and close up when I put zip ties on the balloon's neck [**Figure 7**].

The aquarium tubing was pushed onto the brass tubing, and then the tubing was inserted into the balloon neck and locked on with zip ties. I alternated the position of the zip ties locking nub to decrease the chances of an air leak [**Figure 8**].

The first test used lung power. I found that I didn't have to inhale all that much to create a good vacuum. The coffee became rigid, and I was successful in picking up a variety of objects [**Figure 9**]. However, I found that the coffee grounds also came up through the tube. This did not taste very good [**Figure 9b**]! Next, I attached the aquarium air hose to the 60 ml syringe. I added a zip tie to keep it airtight.

**Figure 10** shows how this rig successfully picked up a variety of objects. I found that I did need to put a fair amount of vertical pressure on the balloon so that it would form around the object and not just sit on top of it.

I now had a working mechanism, so the next step was to automate it.

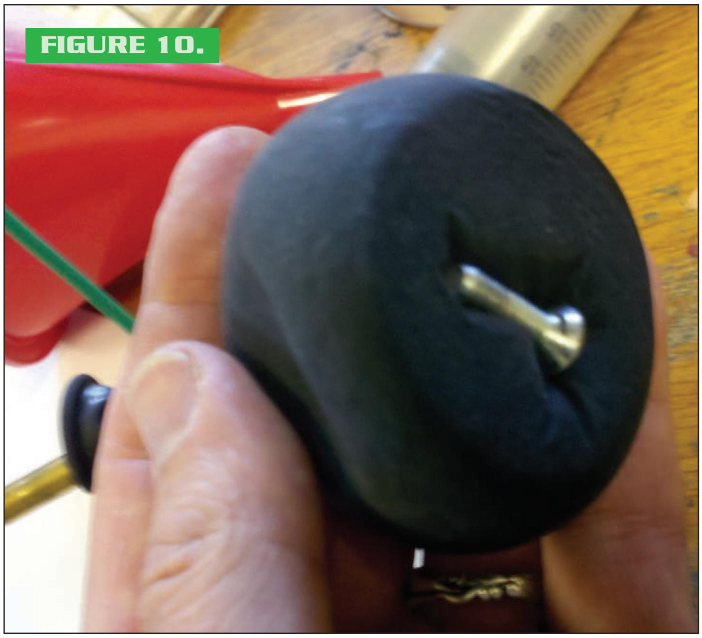


FIGURE 10.

## Automatic Response

I added a funnel to give the balloon and tubing some structure. I took the small funnel I had and used a hacksaw


FIGURE 11.

### Materials

- Good quality latex balloon
- Aquarium plastic tubing
- 60 ml veterinarian syringe
- 11/64" brass tubing
- Zip ties
- Bond 527 cement
- Conductive thread
- Needle
- Craft foam

### Sources

Article in The Proceedings of the National Academy of Sciences
**www.pnas.org/content/early/2010/10/18/1003250107.full.pdf+html**

YouTube video
**www.youtube.com/watch?v=0d4f8fEysf8**

Singularity Hub website
**http://singularityhub.com/2010/10/27/irobots-universal-gripper-pours-a-drink-and-draws-shapes-video**

My video of the gripper in action:
**www.youtube.com/watch?v=_EWWhwSw2IE**

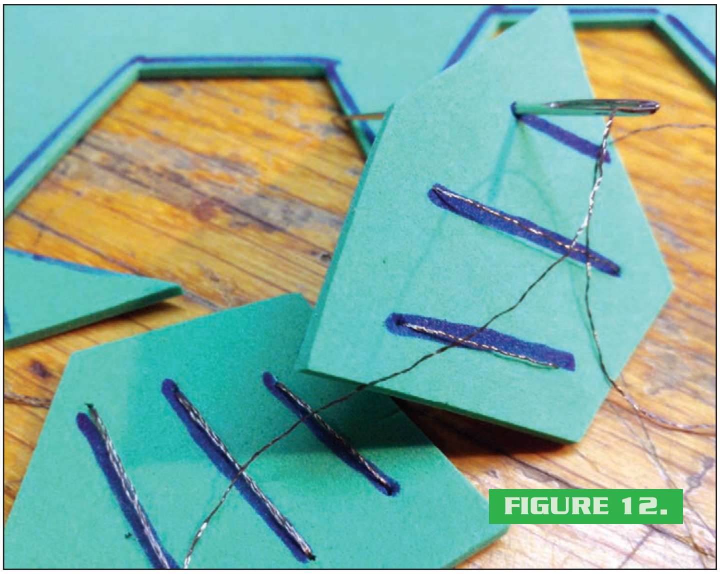


FIGURE 12.

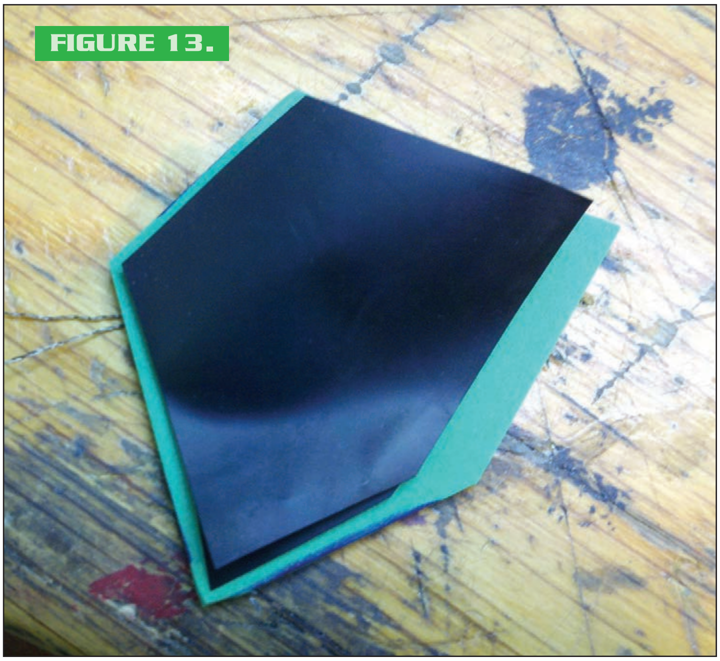


FIGURE 13.

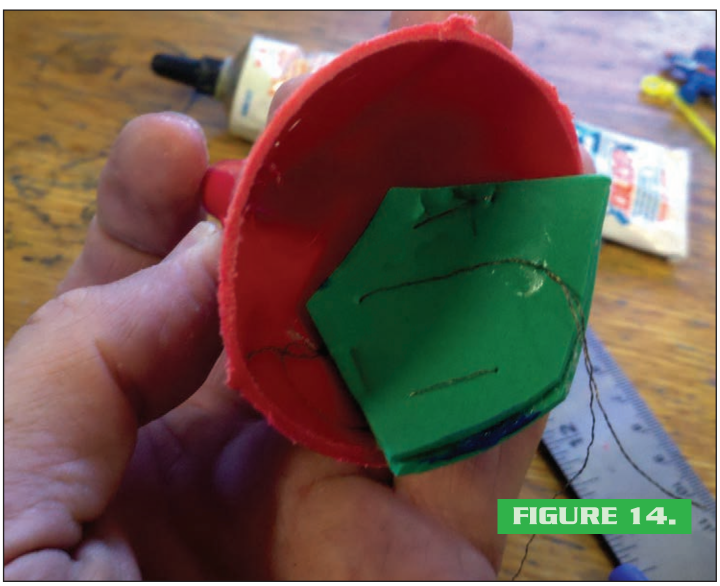


FIGURE 14.

to cut it off shorter. That way, the balloon could hang freely but have support when it was pressed against the object to be picked up. The funnel helped confine the balloon and force the gripper to flow around the object [**Figure 11**].

I decided to put a fabric pressure sensor in the funnel. That way, the Arduino controlling the servo could sense when enough pressure was applied to the balloon. (Plusea also has a great instructable at **instructables.com** describing how to create pressure and bend sensors.) These sensors can be very sensitive and also remarkably accurate in reading pressure or bending.

I created mine out of craft foam from a local craft store. It is flexible, but has enough structure that it doesn't collapse. It is also much cheaper and easier to acquire than the neoprene that Plusea uses. You need a material that can be compressed, but won't stay that way after the pressure is removed [**Figure 12**].

Instead of sewing the edges together, I used Bond 527 multi-purpose cement. It is flexible and also adheres well to the foam. I have found it difficult to stitch together the foam and get just the right amount of pressure. The conductive thread was sewn in a crisscross pattern. The Velostat middle layer was cut smaller than the outer pieces so that the glue did not adhere to it and cause compression. Velostat is available from the EMF safety site at from **www.lessemf.com** [**Figure 13**].

Conductive thread was left sticking out about an inch on both sides to allow connection to a wire running to the Arduino [**Figure 14**].

## Building the Syringe Mechanism

I tested the syringe and found that I only needed to pull the plunger back to 30 ml in order to get a good vacuum. So, I used a hacksaw to cut off the barrel of the syringe. If it was eventually going to fit inside of a robot, I would need it as small as possible. I also cut the plunger off about 3 cm longer than the barrel [**Figure 15**].

I drilled a hole in the servo arm and also through the plunger. A #6 bolt with washers and nuts holds the plunger pretty stable as the servo pulls it back and pushes it forward.

I had some aluminum U channel and cut a piece about 20 cm long. I had a plastic servo-mounting bracket to make a stable platform for the servo. I attached the servo to the plastic mount and then bolted the mount to the U channel aluminum with #6 bolts and nuts.

A piece of flat bar aluminum made the syringe support. The support has to rotate when the plunger is moved. In my mechanism, the syringe must be able to move up and down as the arm pushes or pulls in and out. So, I put one #6 bolt to hold the barrel, and another bolt into the U channel. The U channel bolt was not tightened so that the whole assembly can rotate [**Figure 16**].

Wires from the pressure sensor go to the Arduino. I used the same setup as for a photoresistor on the Arduino site at **www.arduino.cc/playground/**

**Learning/PhotoResistor**. It's an analog sensor, so I used a 10K resistor between the analog input pin and ground. The other side of the sensor goes to +5 volts [**Figure 17**].

Some simple code reads the pressure sensor and when enough pressure is applied, the servo is activated. I've found that each Velostat pressure sensor has its own characteristics, so you just have to experiment with the threshold necessary to sense the pressure when the gripper flows around the object. The Arduino waits two seconds and then pushes the plunger back into the barrel, removing the vacuum. Theoretically, the gripper can hold the object indefinitely, just as long as there are no air leaks in your system [**Figures 18,** and **19**].

FIGURE 15.

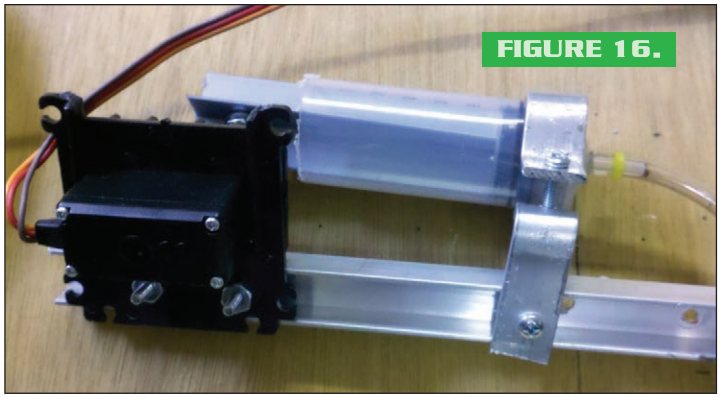


FIGURE 16.

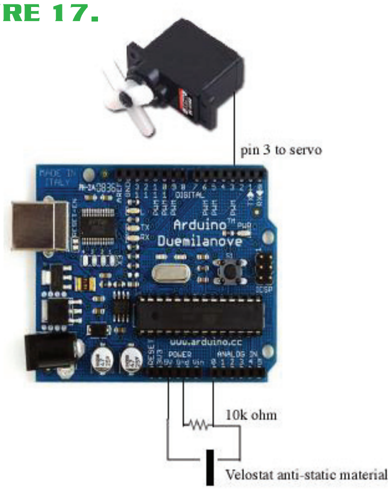
## Getting an Arm Up

I don't have a robot arm (yet), but this could be really fun to put on one. The gripper's strength is that it can pick up very irregular or smooth objects easily. Gripping strength is not affected whether the object is wet or dry. It can also pick up multiple objects with no additional power consumption. Using the syringe, there is no need for vacuum pumps or vacuum reservoirs. The system

**Arduino Code**

```
/*
// Universal Gripper
 // Charles Ford
 // 25 August 2011
 //
 // Sweep
 // by BARRAGAN <http://barraganstudio.com>
 // This example code is in the public domain.
 Simple test of the functionality of the photo
 resistor

 Connect the pressure sensor one leg to pin 0,
 and pin to +5V
 Connect a resistor (around 10k is a good value,
 higher values gives higher readings) from pin 0
 to GND. (see appendix of arduino notebook page
 37 for schematics).

 ----------------------------------------------------

 pressure sensor  10K
 +5     o--/\/\/--.--/\/\/--o GND
 |
 Pin 0 o--------

 ----------------------------------------------------
 */

#include <Servo.h>

Servo myservo; // create servo object to control
               // a servo
               // a maximum of eight servo
               // objects can be created

int pos = 180;       // variable to store the
                     // servo position
int sensorPin = 0;   // define a pin for
pressure
                     // sensor
int pressureValue = 0;    //variable to store
                          //pressure sensor
                          //value
void setup()
{
  myservo.attach(3); //attaches the servo on
                     //pin 9 to the servo object
}


void loop()
{
  pressureValue = analogRead(sensorPin)/2;
// reset value of pressure sensor
// if the gripper is in positive contact
// with a surface
// then activate the suction
  if (pressureValue > 160)
  {

    // apply suction to pick up object
    // servo moves plunger to suction position
    for(pos = 180; pos>=1; pos-=1)
    // goes from 180 degrees to 0 degrees
    {
      myservo.write(pos);
      // tell servo to go to position in
      // variable 'pos'
      delay(10);
      // waits 15ms for the servo to reach
      // the position
    }
    delay(4000);
    // release object
    // push plunger back into the syringe
    for(pos = 0; pos < 180; pos += 1)
    // goes from 0 degrees to 180 degrees
    {         // in steps of 1 degree
      myservo.write(pos);   // tell servo to go
                            // to position in
                            // variable 'pos'
      delay(15);//           // waits 15ms for
the
                            // servo to reach
the
                            // position
    }
  }
  delay(500);
}
```

FIGURE 17.



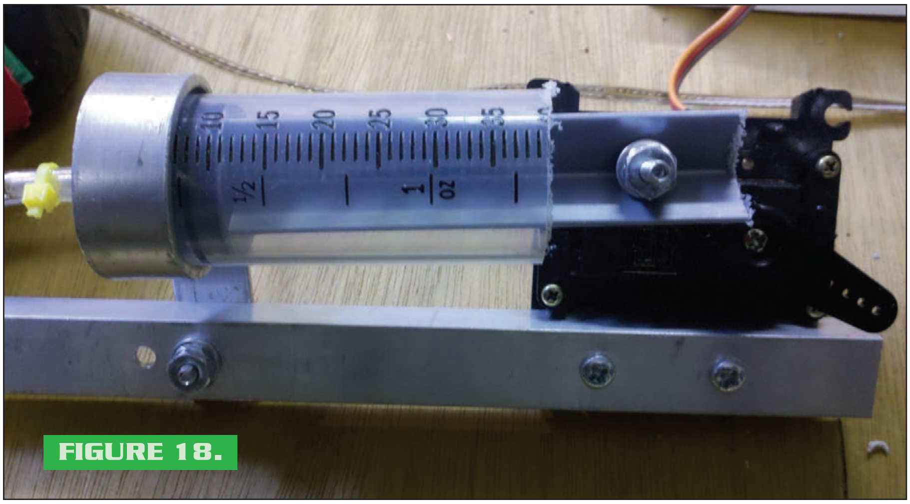



FIGURE 18.

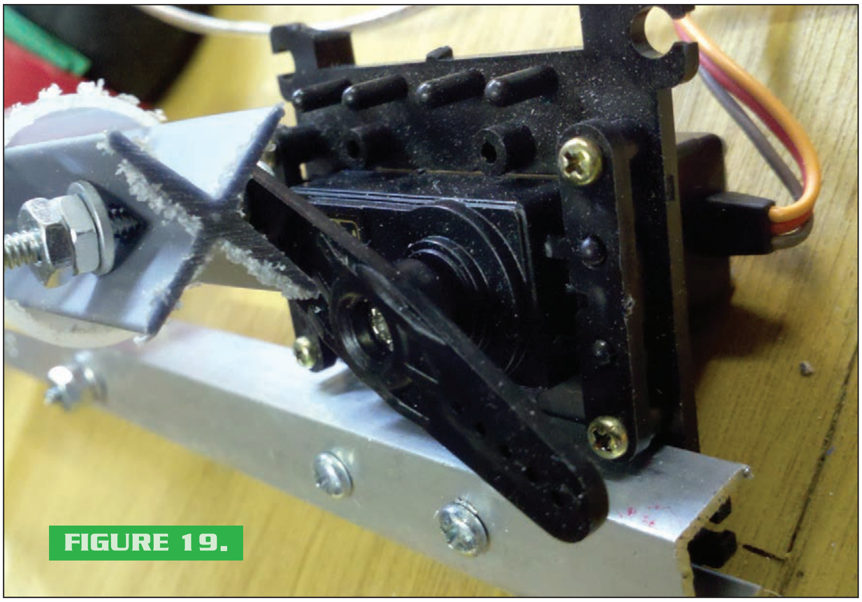


FIGURE 19.

seems robust and reliable. It is also small enough to fit into a robot body without hogging all the real estate.

It does have limitations, however. Because the gripper has to apply pressure against the object, it would not be able to pick up objects from the side. Objects that have no undercuts are difficult to reliably pick up. It cannot pick up flat objects such as paper.

A robotic claw can use pressure sensors to determine that it has made contact with and is holding an object. The universal gripper cannot do that. If an object is too heavy, the elastic balloon may fail.

## I Be Jammin'

Despite the limitations, the universal gripper is another powerful utensil in the roboticist's tool kit. I imagine that we will see it incorporated into robots that must deal with a varied environment; probably in conjunction with more traditional robot hands and claws to allow the maximum versatility.

Jamming may have other uses in robotics, as well. To see a really interesting robot that uses jamming to actually move, you might check out this video: **http://singularityhub.com/2009/12/07/nothing-can-stop-the-blob-bot**.

You can probably see alterations that would improve the design. Let me know if you experiment and come up with improvements! Contact me at fordcharles0@gmail.com. SV

Continued from page 21

as 2.5V, but at the expense of speed. Price is $9.85.

For further information, please contact:

**Solarbotics, Ltd.** Website: **www.solarbotics.com**

## Seven-Channel 2.4 GHz Digital Radio


The popular Eclipse 7 from Hitec has been given a facelift and numerous upgraded features to create the highly anticipated Eclipse 7 Pro.

This advanced AFHSS 2.4 GHz system offers a high intensity white backlit LCD screen and ergonomically positioned, back mounted slider knob and ultra-smooth quad ball bearing gimbals. It possesses 16-model maximum storage, intuitive control switches, and an array of telemetric functions including sensor monitoring and on-board receiver low battery warning.

For further information, please contact:

**Hitec** Website: **www.hitecrcd.com**

## Sub-1 GHz Wireless Connectivity Module

Anaren, Inc., announces a new sub-1 GHz radio frequency (RF) module, based on the CC110L sub-1 GHz value line transceiver from Texas Instruments. Additionally, Anaren has developed TI's RF BoosterPack plug-in board, containing the CC110L-based module which is compatible with TI's MSP430™ microcontroller (MCU) Value Line LaunchPad development kit. The new A110LR09A module-based 430BOOST-CC110L RF BoosterPack, provides electronic devices and equipment designers with a speedy and facile way to develop and test wireless solutions using the popular MSP430 LaunchPad environment. The A110LR09A module is a high performance, dual-band FCC-certified and ETSI-compliant radio module that incorporates TI's CC110L low cost transceiver chip in the industry's smallest package (9 x 16 x 2.5 mm). Operating in the ISM bands at 868/915 MHz, the A110LR09A is well-suited for applications such as sensor networks, industrial monitoring and controls, home and building automation solutions, and remote control toys, among many others. For further information, contact:

**Anaren** Website: **www.anaren.com**

# Meet the Perfect Solderless Breadboard

by Fred Eady

www.servomagazine.com/index.php?/magazine/article/january2012_Eady

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

**The advent of the solderless breadboard brought about the decline of breadboarding circuitry using point-to-point techniques that involved perfboard, wire, and solder. Over time, solderless breadboards found their way onto printed circuit boards (PCBs), acting as universal grazing areas for plug-in connections and leaded electronic components.**

Before moving on to SMT devices and specialized PCBs, I used solderless breadboards to permanently support often used circuit modules such as RS-232 interfaces and seven-segment displays. Previous to becoming a full-time author, I worked as a design consultant for a telephone company. I kept a work-in-progress solderless breadboard that morphed differing telephony circuit configurations as the customer's requirements changed. After becoming a writer, many a design was tested on a solderless breadboard before being committed to a PCB.

As useful as solderless breadboards are, when you get right down to it they are dumber than rocks. After you plant all of the necessary electronic components on a solderless breadboard, you still need to supply power externally and electrically attach the necessary debugging and monitoring equipment. Wait. You're not done. Unless your application only blinks LEDs, you'll most likely need to write a companion test application to interface to your solderless circuit. That average test application will need to provide stimulus to your solderless circuit and react to the applied stimulus in a predetermined manner. If your circuit fails to respond as expected, that logic probe, voltmeter, oscilloscope, and signal generator you have standing by may have to be called into action.

In the days of the dinosaur, the oscilloscope was the size of a Volkswagen and all of that stimulus circuitry would be sharing space on the plains of the solderless breadboard. The crude test application would be a hastily thrown together Visual Basic application that communicated with the solderless circuit serially or via a specialized printer port digital interface.

Dinosaurs no longer roam in large numbers. Oscilloscopes aren't built around a cathode ray tube. And, .NET and Arduino are the new programming paradigms. This is 2012 and the solderless breadboard has finally become an intelligent part of the electronic hardware design process.

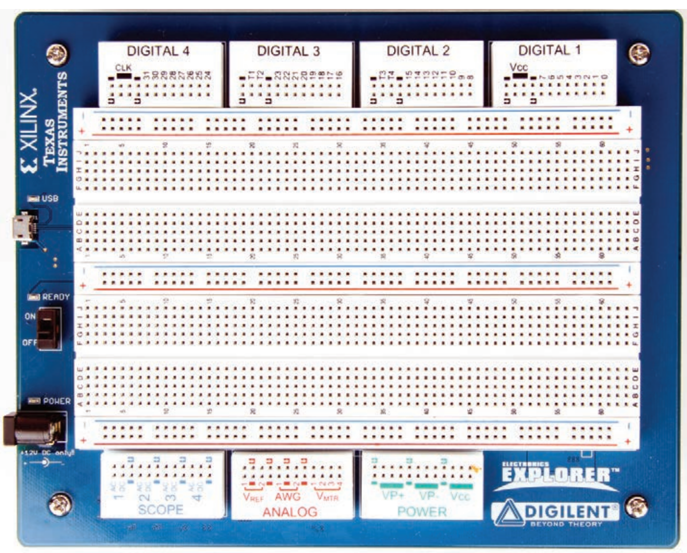



**PHOTO 1.** This is just another solderless breadboard ... until you turn it over.

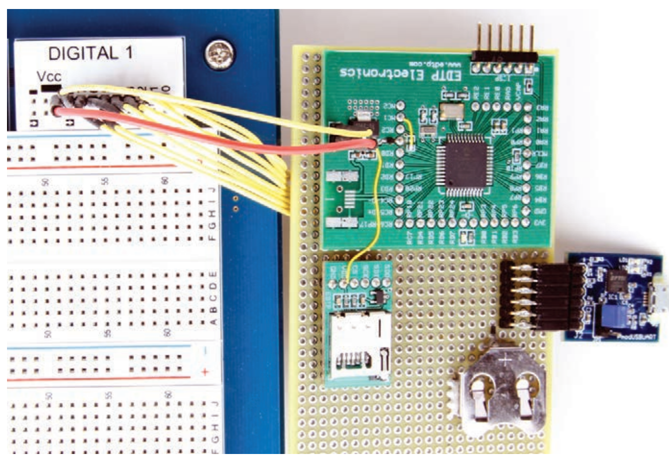
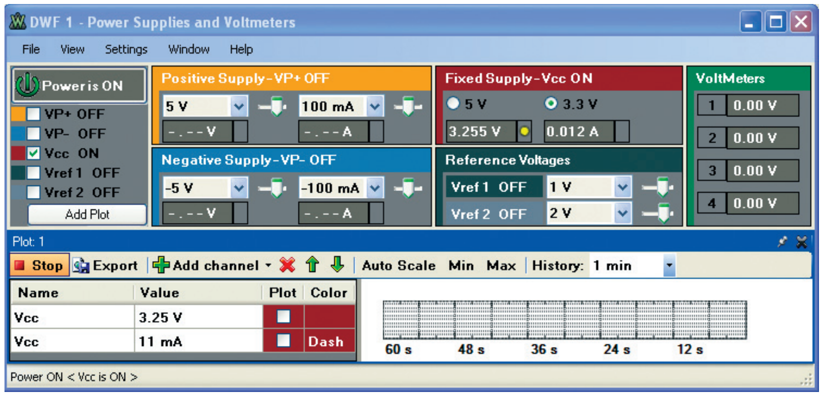



**PHOTO 2.** The Digilent Electronics Explorer is designed to be an analog/digital instrument package. As far as the PIC18F46J13 is concerned, the Explorer is a bunch of LEDs and pushbuttons. It is also the PIC18F46J13's power supply. We can even power a PICkit3 debugger from the Explorer's fixed supply.




**SCREENSHOT 1.** I've turned off the reference power supplies and selected the 3.3 volt fixed supply to power the PIC18F46J13 microcontroller and PICkit3 debugger.

## The Perfect Solderless Breadboard

What may just be the perfect solderless breadboard is smiling for the Canon in **Photo 1**. The solderless breadboard area is augmented by special-purpose solderless breadboard bricks. The bricks expose four voltmeters, digital I/O, external trigger inputs, a four-channel oscilloscope, an arbitrary wave form generator, a pair of reference voltages, and positive/negative supply voltages.

## Static Digital I/O

Four of the solderless breadboard bricks contain 32 digital pins that are shared with the Digilent Electronics Explorer's Logic Analyzer and Digital Signal Generator modules. The 32 digital pins are virtually linked to a USB-attached PC computer as pushbuttons, LEDs, switches, sliders, seven-segment displays, and progress bars.

The Electronics Explorer is heavily oriented towards analog exploration. However, the Electronics Explorer is perfect for those that want to work with microcontroller circuitry. Utilizing the Electronics Explorer's Static Digital I/O module eliminates the need for physical LEDs and pushbutton switches.

In **Photo 2**, I attached a PIC18F46J13-based microcontroller board to the Electronics Explorer's Digital 1 solderless breadboard brick. The Digital 1 brick exposes the least significant bits of the Explorer's 32-bit Static Digital I/O module. The low-order solderless breadboard brick also contains Vcc and ground points based on the Explorer's 5.0 volt and 3.3 volt power supplies. As you can see in **Screenshot 1**, the Explorer is powering the PIC18F46J13 board with 3.3 volts. Its Fixed Supply has enough current available to also power a PICkit3 debugger/programmer.

The PIC18F46J13's I/O port D is physically attached to the Digital 1 solderless breadboard brick. Let's use the CCS C Compiler and a PICkit3 to force the PIC18F46J13 to manipulate the Digital 1 brick as if it were a bank of LEDs. To accomplish this on the Explorer side of the river, we must configure its least significant eight static I/O bits for bit I/O. Right-clicking on each of the least significant eight bits in **Screenshot 2** allows us to choose to view the bits as LEDs.

Obviously, the virtual LEDs in **Screenshot 2** are changing state. In reality, the virtual LEDs are displaying the state of the PIC18F46J13's PORT D which is programmed as a binary up counter. Excerpts from the MPLAB X *electronics-explorer.c* source file back up my claim:

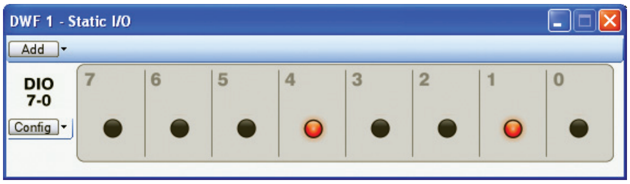
```
#define real_led            PIN_E0
unsigned int8 counter;
void init(void)
{
 set_tris_d(0b00000000);
   //connected to Digital 1 of the
   //Electronics Explorer
 output_d(0b00000000);
 set_tris_e(0b00000000);
   //drives a real LED on bit 0
 output_e(0b00001111);
 lastsec = servo_time.tm_sec;
 counter = 0;
}
 void main()
{
     init();
     do {
          rtc_read(&servo_time);
          if (servo_time.tm_sec != lastsec)
          {
               output_toggle(real_led);
               ++counter;
               output_d(counter);
```

The preceding code snippet is obviously incomplete. However, you can see that a real LED is toggled every second according to the time kept by the PIC18F46J13's internal RTCC. Following the action on the real LED, the eight-bit *counter* variable is incremented and presented on PORT D of the PIC18F46J13. The state of the PORT D I/O pins is represented in **Screenshot 2**.

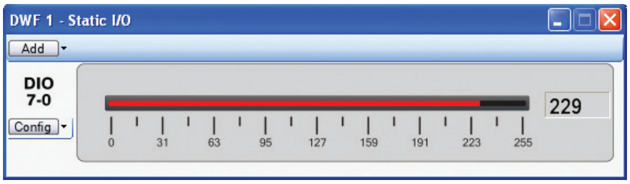
Let's use that same LED binary counter code to drive another Explorer indicator. This time, we'll configure the Digital 1 brick as a progress bar. As the count on the PIC18F46J13's PORT D increases, the progress bar in **Screenshot 3** is updated accordingly.

Unless you have access to a preassembled array of seven-segment displays on a PCB, you'll have to wire up that array pin by pin. What a pain! Wiring up virtual seven-segment displays is much easier. In fact, we've already done that. **Screenshot 4** is the result of configuring the Digital 1 brick as a seven-segment display. We can use the binary counter code to illuminate each segment associated with the binary count. That wouldn't be of any use, however, I've seen professional instruments that randomly illuminated segments of a seven-segment display to indicate activity. Since we have some binary counter code written, let's translate the binary on PORT D to a human-readable 7-segment character using caveman coding. A caveman would begin this coding by defining all of the valid hexadecimal and numeric bit patterns that count from 0x00 to 0x0F:

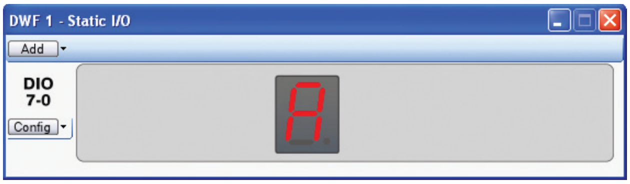
```
#define dp   0b10000000      //decimal point
#define d0   0b00111111
#define d1   0b00000110
```



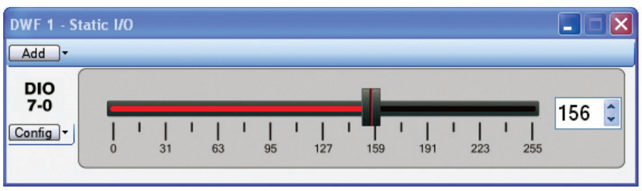

**SCREENSHOT 2.** The Electronics Explorer's virtual LEDs do not require current-imiting resistors. Better yet, a single connection between the microcontroller I/O port and the Explorer is all that's needed to use them.




**SCREENSHOT 3.** This Electronics Explorer instrument is yet another way to display the state of the PIC18F46J13's PORT D I/O pins. Decimal 229 is equivalent to 0xE5 or 11100101 binary.




**SCREENSHOT 4.** There are many ways to convert the seven-segment bit patterns to human-readable form. Here, a caveman shows us how he would do it using the C switch mnemonic and some bit fields.




**SCREENSHOT 5.** As you can see, I moved the Explorer virtual slider from the decimal 94 position to the decimal 156 position.

```
#define d2    0b01011011
#define d3    0b01001111
#define d4    0b01100110
#define d5    0b01101101
#define d6    0b01111101
#define d7    0b00000111
#define d8    0b01111111
#define d9    0b01101111
#define da    0b01110111
#define db    0b01111100
#define dc    0b00111001
#define dd    0b01011110
#define de    0b01111001
#define df    0b01110001
```

After slamming a couple of rocks with his club, the caveman coder would associate each seven-segment bit pattern with the caveman-readable character:

```
++counter;
switch(counter)
{
  case 0x80:
    output_d(dp);
  break;
  case 0:
    output_d(d0);
  break;
  case 1:
    output_d(d1);
  break;
  case 2:
    output_d(d2);
  break;
  case 3:
    output_d(d3);
  break;
  case 4:
    output_d(d4);
  break;
  case 5:
    output_d(d5);
  break;
  case 6:
    output_d(d6);
  break;
  case 7:
    output_d(d7);
  break;
  case 8:
    output_d(d8);
  break;
  case 9:
    output_d(d9);
  break;
  case 0x0A:
    output_d(da);
  break;
  case 0x0B:
    output_d(db);
  break;
  case 0x0C:
    output_d(dc);
  break;
  case 0x0D:
    output_d(dd);
  break;
  case 0x0E:
    output_d(de);
  break;
  case 0x0F:
    output_d(df);
  break;
}
```

**Screenshot 4** is a representation of the hexadecimal value 0x0A. Not bad for a caveman.

My Mom always says, "Turn about is fair play." So, let's configure the PIC18F46J13's PORT D as a digital input port. Then, let's configure the Explorer's Digital 1 solderless breadboard brick as a digital output port in the form of a slider. We'll use HyperTerminal and the PIC18F46J13's serial port to display the value presented to PORT D. Again, this is so easy a caveman can do it:

```
printf("%X\r\n",input_d());
```

After disabling the binary counter and time display routines, that's all we have to add to our existing code. The results of our folly are depicted in **Screenshot 5**. Pretty neat, huh? The Explorer's logic analyzer is always connected to the Digital bricks as an input device. What you're witnessing in **Screenshot 6** is being generated by this CCS C built-in command:

```
output_d (counter++);
```

Note that I've configured the logic analyzer to record on every transitional edge of bit 0. I'm also capturing the bit patterns in a tabular form that can be exported to programs like Microsoft Excel for analysis.

**Screenshot 7** represents the addition of static I/O bit 31 as a 1.0 kHz clock source. Note that the added clock source was added to the logic analyzer view. Also note that the virtual LEDs are doing their thing despite the clock on bit 31.

## This Gets Even Better

Check out **Screenshot 8**. I've converted the least significant bits of the Digital 1 brick from LEDs to various types of switches. These switch outputs can be used as logic elements to drive the PIC18F46J13's I/O pins. The most obvious switch is the Explorer's pushbutton switch which is shown in action in **Screenshot 8**. I've used this line of code

Digilent Electronics Explorer
**www.digilentinc.com**

to read the state of each switch:

```
printf("%X\r\n",input_d());
```

You can see the Digital 1 brick's bits represented in hexadecimal in the left sector of the terminal emulator Screenshots.

The schematic depiction of the pushbutton tells it all. The equivalent circuit is a simple SPST switch providing a logical high to the pulled down Digital 1 I/O pin D0. Be careful and don't get lulled into a state of false security. The virtual pushbutton transitions smoothly between logic levels. If you're going to port over to a real mechanical switch, you'll need to debounce the real switch in your final application.

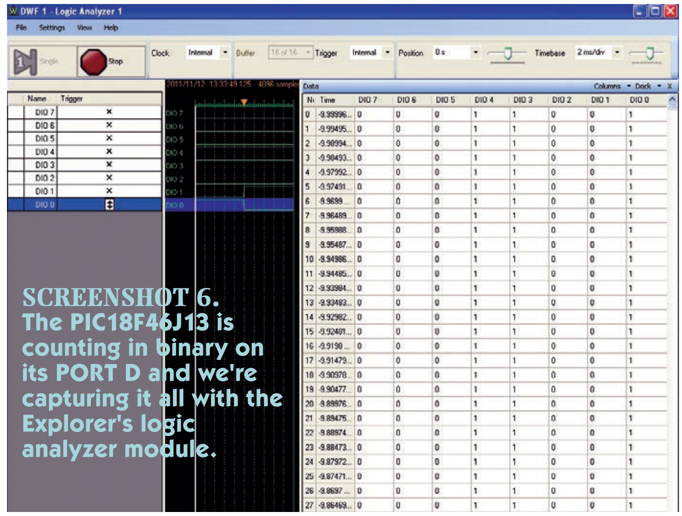
The Push/Pull switch featured in **Screenshot 9** takes the pushbutton concept a step further. You can line up a maximum of 32 push/pull switches to simulate logic levels that could be presented to the PIC18F46J13's I/O pins. The push/pull switch configuration could also be used to supply logic levels to logic gates and multiplexers.

An eight-bit Transparent Octal Latch (74LS573) can be simulated by deploying banks of three-state switches. The three-state switches can be positioned to present a logically high, logically low, or high impedance state. When the three-state switch is at the Z position, other logic signals attached to D2 are free to operate without any logical interference.

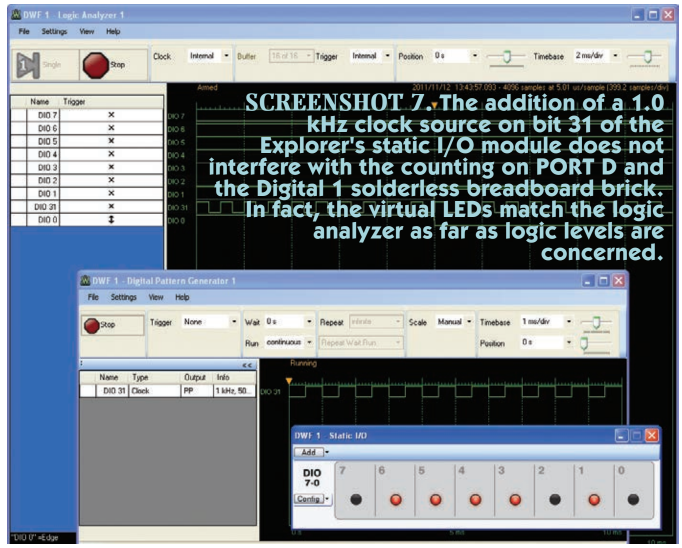
The final set of switches can be visualized as MOSFET devices configured as open source and open drain. Let's look at the open source switch that is configured on I/O bit 3 in **Screenshot 11**. When the open source switch is in the Z position, the drive signal is removed from the P-channel MOSFET's gate. At this point, the PIC18F46J13's I/O pin being serviced by the Explorer's D3 I/O pin is free of any logical influence from the open source switch. Applying the logic of the schematic snippet in **Screenshot 11** tells us that providing a drive signal to the MOSFET gate will result in presenting 3.3 volts to the PIC18F46J13 I/O pin. In the case of the P-channel MOSFET, the drive signal is logically low. However, the result at static I/O pin D3 is identical and a logic high level is presented by the D3 pin. This P-channel MOSFET switch concept is commonly used to form a logic controlled solid-state power switch.

The operation of the last switch in the set should be familiar to those of you that have used logic level MOSFETs to drive higher voltage devices. The open drain switch is identical to an N-channel MOSFET with its source grounded and its drain open. Digital 1's I/O pin D4 is set for high impedance operation in **Screenshot 12**. Judging from the hex readout, nothing is driving the N-channel MOSFET's gate and the D4 pin is left to float. Switching to the zero position applies a logical high drive signal to the N-channel MOSFET definitely turning it on. Turning the MOSFET on pulls D4 to ground level. What do we have to do to get a logic high at the MOSFET's drain?

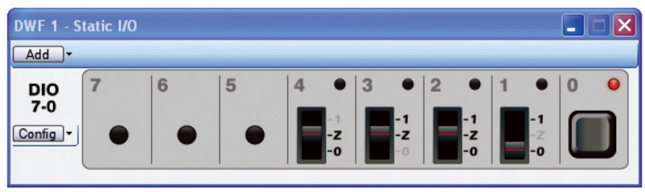
What would happen if we drove the floating MOSFET drain with a positive voltage? We don't want to apply 3.3 VDC directly to the drain because that will release the MOSFET's magic smoke when it is turned on. So, we'll insert a resistor between the applied voltage and the MOSFET gate. That's easy enough to do. So, I've physically pulled up D4

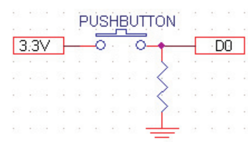



**SCREENSHOT 6.** The PIC18F46J13 is counting in binary on its PORT D and we're capturing it all with the Explorer's logic analyzer module.

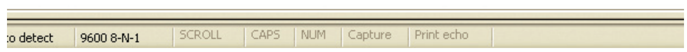


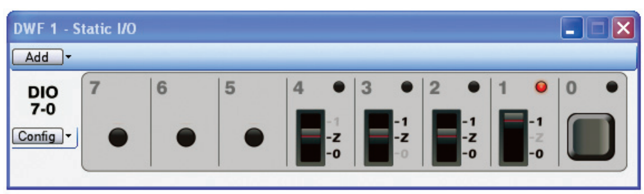

**SCREENSHOT 7.** The addition of a 1.0 kHz clock source on bit 31 of the Explorer's static I/O module does not interfere with the counting on PORT D and the Digital 1 solderless breadboard brick. In fact, the virtual LEDs match the logic analyzer as far as logic levels are concerned.

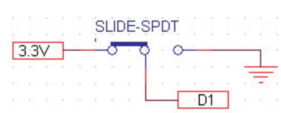


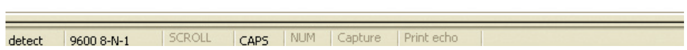

**SCREENSHOT 8.** From left to right the switches are open drain, open source, three-state, push/pull, and pushbutton. Think of these switches as logic sources instead of switches.



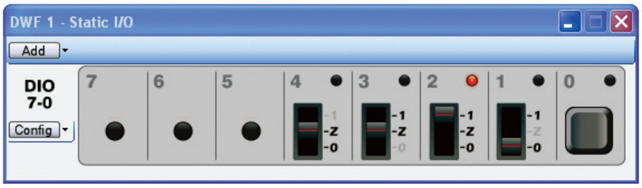



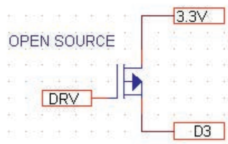

**SCREENSHOT 9.** Call upon the push/pull switch configuration when you need a sticky logic level.

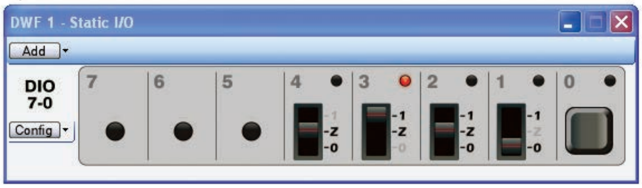


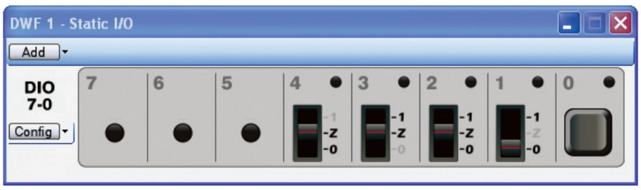
**SCREENSHOT 10.** This switch emulates an eight-bit transparent octal latch. When put into a high impedance state, the I/O pin relinquishes all control of the logic signal attached to it.

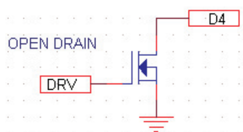

**SCREENSHOT 11.** Although the P-channel MOSFET logic differs from the perceived operation performed by the open source switch, the outcome is identical. The open source's active output is logically high, which is the 1 switch position.



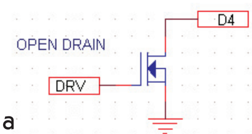



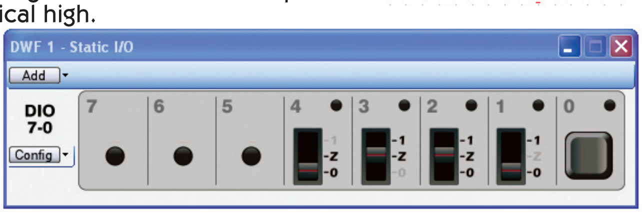




**SCREENSHOT 12.** This is about as dead as a circuit can be. D4 is attached to a PIC18F46J13 high impedance input pin and the N-channel MOSFET gate is floating.



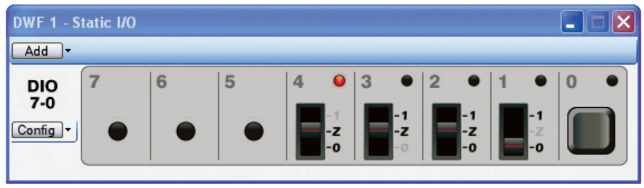

**SCREENSHOT 13.** Turning on the N-channel MOSFET gate brings the MOSFET drain to ground level. Right now, we don't have a way of getting the MOSFET drain to present a logical high.



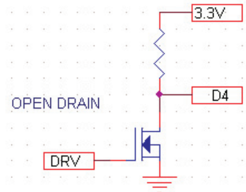



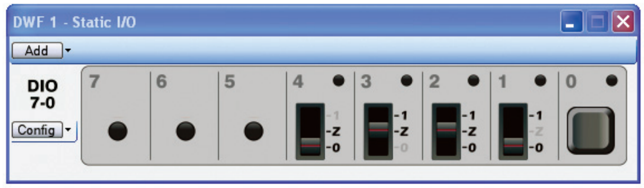

with the addition of a real resistor between D4 and Vcc. Bang! The D4 status LED illuminates in **Screenshot 14**.

With the open drain switch in the Z position, the N-channel MOSFET is turned off and D4 assumes the +3.3 volt logic level through the pull-up resistor. Moving the open drain switch to the zero position turns the MOSFET on and draws D4 to the ground level, producing a logical low output at D4.

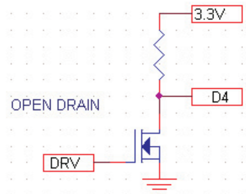



**SCREENSHOT 14.** It's pretty obvious how the open drain circuit works. The open drain's active output is logically low, which is the zero switch position.




**SCREENSHOT 15.** This circuit is commonly used to drive LEDs and high voltage/high current relays using a logic level. The circuit also can be used as a simple logic inverter.



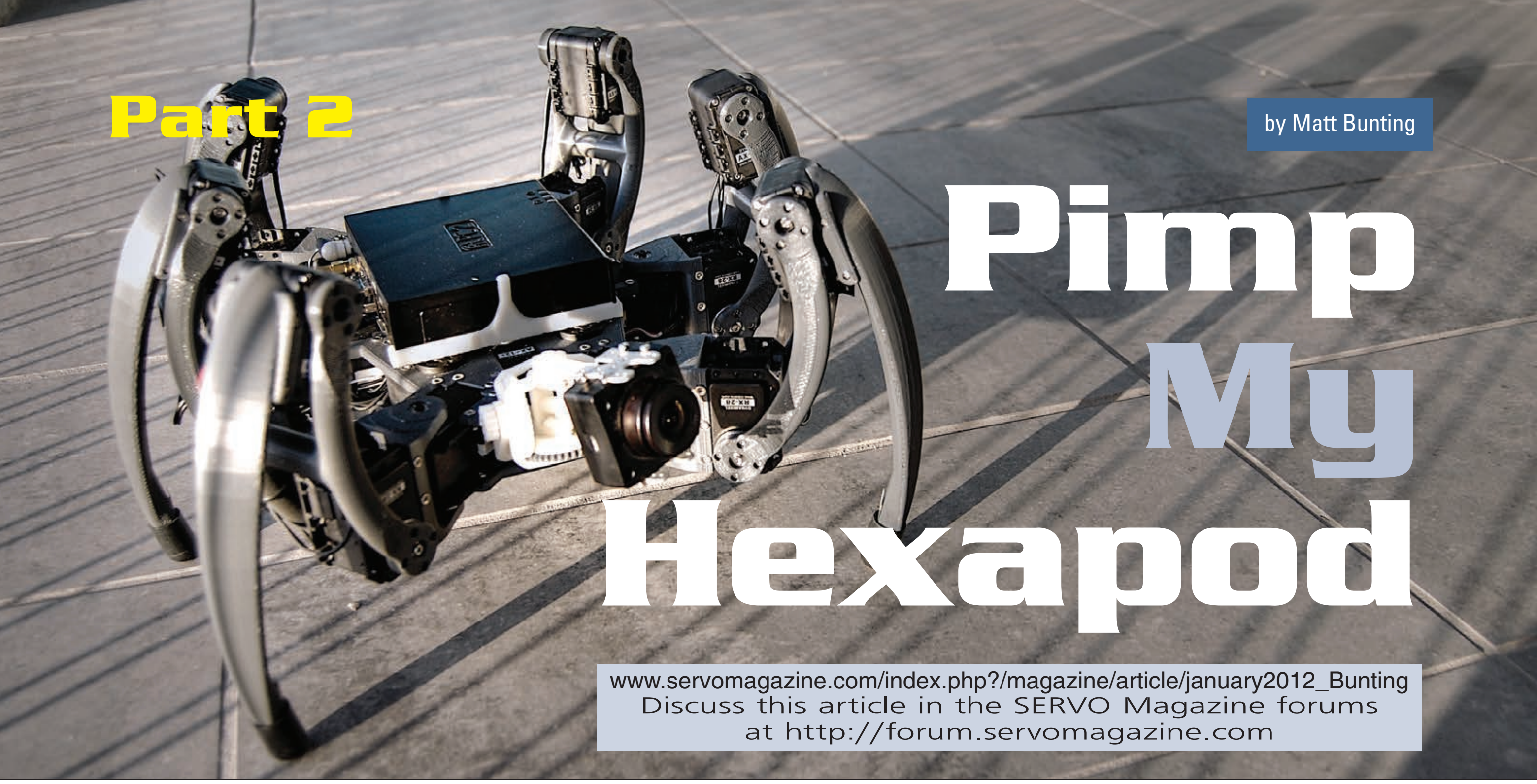

The on state of the MOSFET is supported by the extinguished status LED in **Screenshot 15**. The switch logic seems flipped. However, it makes perfect sense. The open source switch output is active when placed in position 1. That's because the active state of the open source circuit is logically high. The open drain switch is active when placed in the zero position. Thus, the open drain switch active output is logically low. Think of it this way. The open source and open drain active outputs follow the logical output conditions of the P-channel and N-channel MOSFETs when the MOSFETs are turned on. The open source circuit in **Screenshot 11** outputs a logical high on D3 when the P-channel MOSFET is turned on. The open drain circuit in **Screenshot 15** outputs a logical low when the N-channel MOSFET is turned on. Piece of cake.

# Disneyland

Did you know that all of the illusions at Disneyland are done above ground while all of the reality that supports the illusions lies underground? The Electronics Explorer uses a similar concept except that what happens on the Explorer's solderless breadboard is no illusion. The reality on the solderless breadboard is produced by the multiple power supplies and logic devices on the opposite side of it.

In actuality, the world of robotics is a complementary mix of analog and digital techniques. I've presented the Digilent Electronics Explorer in the role of a microcontroller stimulus generator and virtual logic monitor.

The Explorer is just as flexible when used as an analog workbench. In that the Explorer easily mixes digital and analog experimentation and project development, it is a perfect tool for any robot head. **SV**



Part 2

by Matt Bunting

# Pimp My Hexapod

www.servomagazine.com/index.php?/magazine/article/january2012_Bunting
Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

**This time, we'll explore and implement two upgrades for the hexapod. The first revision involves transitioning from the Dynamixel RX-28 motors to the MX-28 motors. Some of the new features of the MX-28 are demonstrated and tested. The second upgrade involves a better solution for the head gearing system. The involute curve is explained and then implemented in 3D printed gears.**

## Introduction

Previously, I had discussed some of the various issues with my current hexapod design. After going through the current design and past upgrades, some of the breaking points were explored. Every time I go through an upgrade process, most parts show some signs of wear and tear.

Some parts break much faster than others. Here, I will cover two parts from opposite ends of the spectrum. The RX-28 took two years of a lot of use before some of the parts broke down. The head mechanism took a month before the gear teeth broke.

I will review the new Robotis MX-28 motor and compare it to the RX-28. The newer motor offers some of the latest features that most servo motors wish they had. Having heavily used the RX-28 motors, I'm sure that I will welcome all the new features.

The gears in the head mechanism were never properly designed, and therefore did not last very long. Though there are proper ways to design gears, some of the methods are a little difficult to understand. The involute shape itself may be easy to understand, but it's not necessarily clear as to why the shape works as well as it does.

## Motors

The RX-28 motors have been more than exceptional to work with, providing great features such as being able to read the motor load, current position, motor temperature, input voltage, plus implement emergency shutdown and configure the control system. These features can be essential to both professional and hobby roboticists. Before exploring the control features of the MX-28, let's first compare it with the RX-28.

## MX-28 vs. RX-28

Each MX-28 comes in a nice package with plenty of hardware. **Figure 1** shows the contents of each kit. Each motor kit includes a motor, servo horn with shaft bolt, bolts for mounting to the servo horn, nuts and bolts to mount to the body of the servo, and a cable for power and communication. The kit is very similar to the RX-28 motor kits, but there are some notable differences. The white washer shown in the photo is actually a thrust washer. This helps to prevent off-axis torque from tweaking the servo horn too much. The cable included is also only a three-pin, not four. Lastly, this kit does not include a molex connector to build your own cable, nor does it come with the mating connector to solder onto your PCB. The thrust washer is a nice addition, but I did like how the RX-28 motors included the additional cable components.

**Figure 2** shows the RX-28 and MX-28 side by side. The

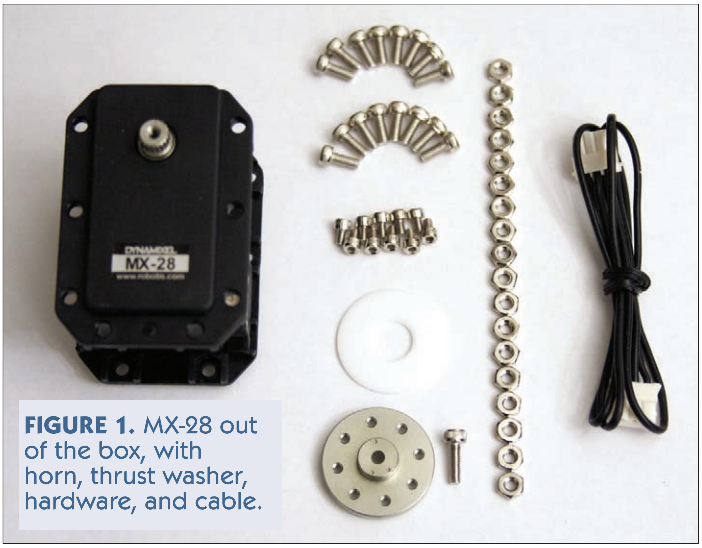



**FIGURE 1.** MX-28 out of the box, with horn, thrust washer, hardware, and cable.

main body of the servo is identical except for the electrical connector is three-pin instead of four-pin. The version of the RX-28 that I used in the hexapod was purchased a few years ago, so the servo horn was a bit different style. The threaded servo horn holes have actually been drilled on a slightly different radius. The newer horn style has threaded holes on a 16 mm diameter, whereas the older horn style has holes on a 17 mm diameter. When upgrading, it is important to make note of this as any brackets will need to be redesigned or drilled out.

Fortunately, this issue is fairly minor compared to the different communication bus. The serial control protocol is still identical, however, Robotis switched the protocol from the RS-485 bus to bidirectional half-duplex TTL communication. This transition is nice in that it reduces the overall wiring which will definitely be helpful when trying to cram all the excess wire inside the hexapod body. I believe the TTL bus also consumes a bit less power than the RS-485 bus. The USB2Dynamixel supports the TTL bus as well, and now the MX-28 can be daisy-chained directly to the AX-12 or AX-18 actuators.

Unfortunately, the half-duplex TTL bus is not as well supported outside of Robotis. Before, it was possible to get any FTDI based USB to RS-485 adapter and very quickly run some Linux code and begin communicating. Without the USB2Dynamixel, a method of tristate buffers (like chips 74HC125 or 74HC126) must be implemented to communicate through a standard UART or FTDI chip. It is by no means a difficult circuit to build; it is just not quite as plug and play as before. It's nice to see that Robotis appears to be making a move to have all their motors use the same protocol. Anyone already set up with the AX-series will definitely be happy with the new transition. I would think that when the MX-64 and MX-106 come out they may also have the TTL bus.

## Inside the MX-28

Opening up both the MX-28 and RX-28, we can see a bit of what is going on internally. In **Figure 3**, the RX-28 is on the left, showing the L6201 H-bridge. On the right is the MX-28, showing the 72 MHz STM32F103C8T6 ARM Cortex M3 processor — which is quite a step up from the ATMega8 in the RX-28. Both motors include a spot for a programming header, enabling any hacker to quickly load up their own code. **Figure 4** shows the opposite side of the MX-28 board. Here, the 74HC126 tristate buffer is visible, as well as the AS5045 magnetic potentiometer. Still in the body of the servo is the magnet attached to the output shaft. Clearly, the old slot for the potentiometer used in the RX-28 still exists, meaning that the body is most likely interchangeable with the RX-28. **Figure 5** shows the gearbox of the MX-28.

The magnetic sensing is what puts the M in MX-28. This is the most exciting part for implementing the motor in the hexapod as it will never wear out since there is no mechanical contact. The AS5045 uses a set of Hall-effect sensors to determine the orientation of the magnet. The previous continuous turn potentiometers eventually became worn after running the hexapod for an inordinate amount of time. The magnetic sensing also extends the angle control range to a full 360 degrees (from 300) at a 12-bit resolution (from 10-bit). This increases the precision from .293 degrees to .0879 degrees. That is a total factor of 3.33 times better.

The new ARM processor is able to bump up the communication bus to include three newer baud rates at 2250000, 2500000, and 3000000. I will definitely be implementing the newer baud rates which will make the hexapod operate as a much smoother machine.


**FIGURE 2.** The MX-28 and RX-28 motors are nearly identical.

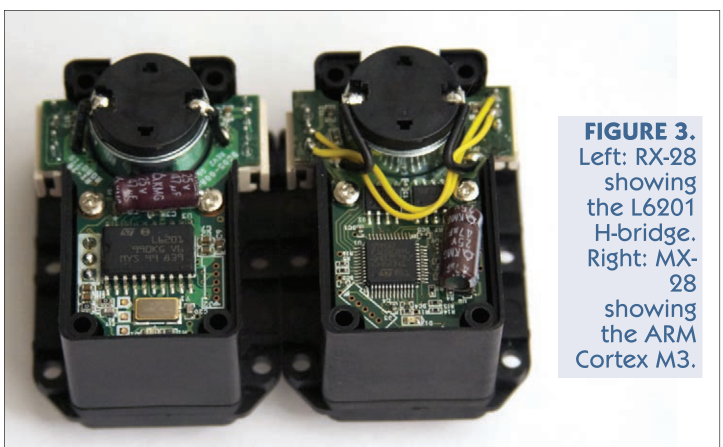
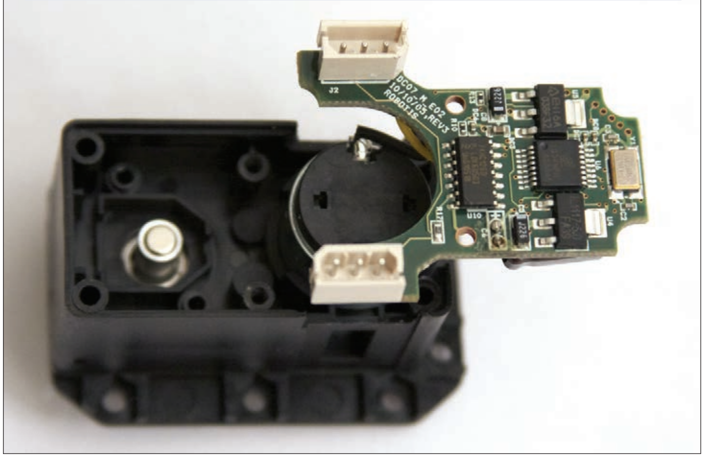


**FIGURE 3.** Left: RX-28 showing the L6201 H-bridge. Right: MX-28 showing the ARM Cortex M3.

**FIGURE 6.** Motor test setup using the hexapod body, half of a femur, tape, and a solid metal weight.

**FIGURE 4.** Flipping the board around reveals the magnet and 12-bit 360 degree AS5045 magnetic potentiometer.



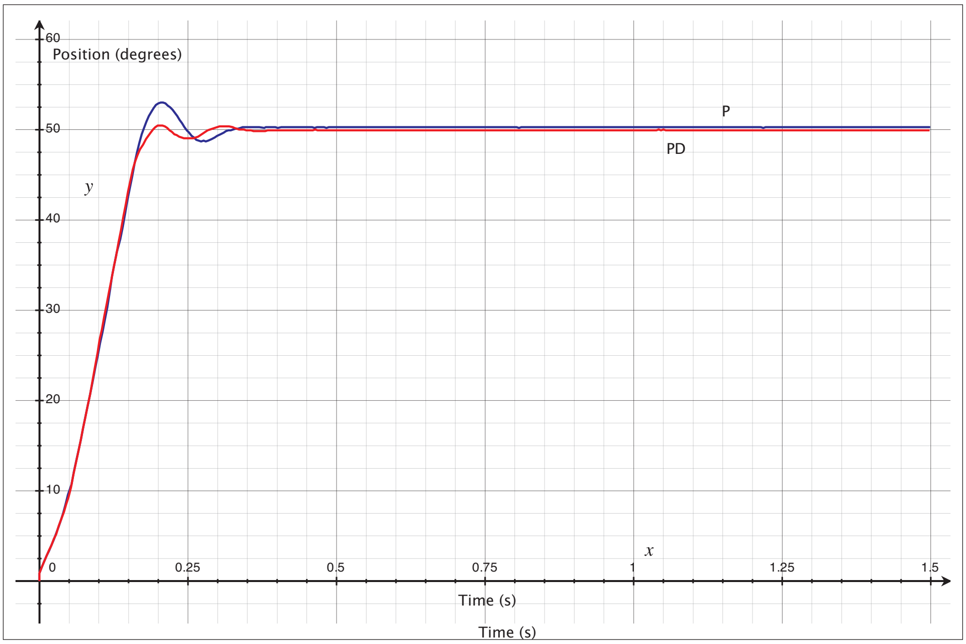



**FIGURE 7.** P and PD comparison. The pure P controller (red) overshoots the goal position of 50 degrees, but the implementation of the derivative feedback term in the PD controller (blue) reduces the overshoot. Both controllers still exhibit an error from the lack of an integral term.

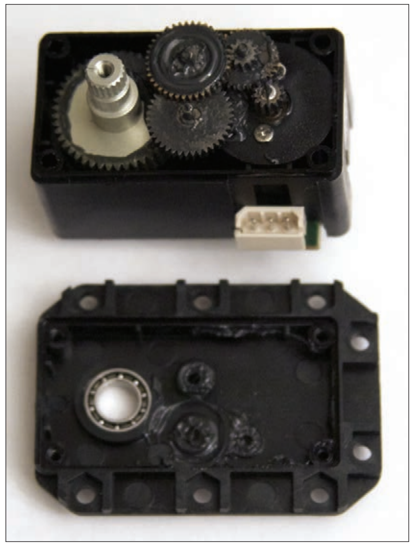


**FIGURE 5.** The gearbox of the MX-28, very similar to the RX-28.

## Control System

As previously discussed, though some control systems may work well for different scenarios, no one control system can work in all situations. The control system used in the RX-28 involved setting a slope and certain limits. Though easy to understand, the controller was non-linear and difficult to adjust for most robotics projects. Under the right tuning, the motor could be of good use for human-robot interaction, where it is desirable to have the motor move quickly to a position but still remain compliant when getting physically moved in unpredictable ways.

The PID controller is much more useful when operating many systems, especially those that can be modeled as a mass-spring-damper system. **Figure 6** shows a simple setup with the motor strongly clamped to a table, with half of a hexapod femur and a heavy iron weight taped to the end.

### LINKS

My Personal Website
**www.12centdwarf.com**

Robotis
**www.robotis.com**

Intel Embedded
**http://edc.intel.com**

Robot Source
**www.robotsource.org**

Dimension 3D Printers
**www.dimensionprinting.com**

Involute Gear Design
**www.cartertools.com/involute.html**

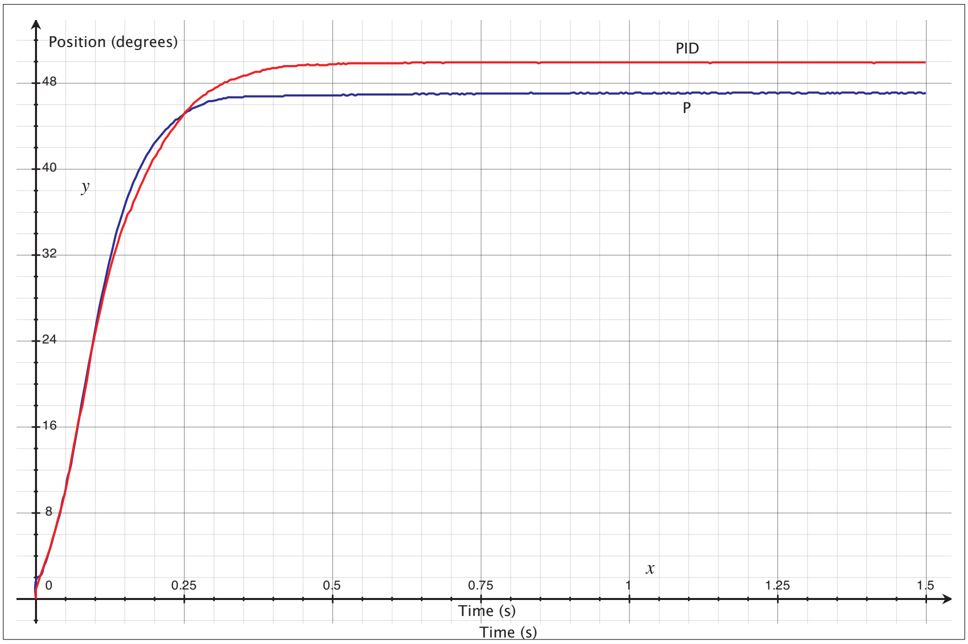



**FIGURE 8.** P and PID comparison. Here, the P controller is set at a very low gain and struggles to get within three degrees of the goal position of 50 degrees. The integral term in the PID controller tightens up the output shaft right at 50 degrees.

The setup is crude, but will demonstrate some of the functionality of the controller. The heavy mass acts as, well, the mass; the tape operates as a spring; and the thrust washer acts as the damper. A program was written in Ubuntu to initially set the motor at the zero position, set the gains, then set the motor to a position of 50 degrees. Immediately after, the program reads the motor position internally and outputs the position and time to a file, where the data can then be plotted.

Two different comparison tests were executed, comparing a standard P controller to a PD, and then a PID controller. **Figure 7** shows a P controller where the gain is set fairly high. This results in an overshoot of the output angle. By implementing a small amount of a

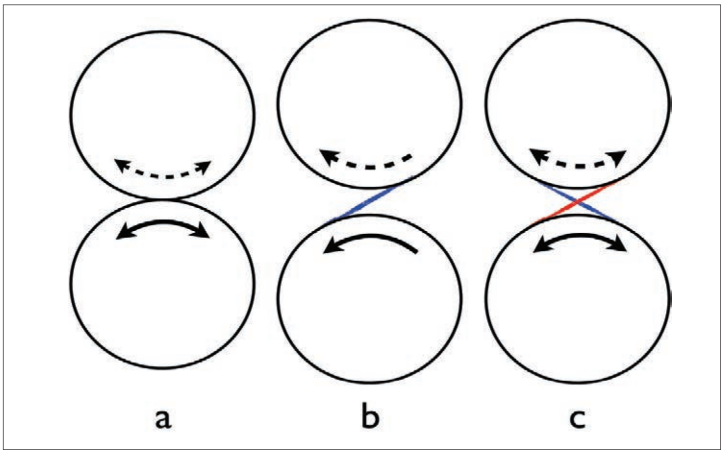



**FIGURE 9.** Gear concepts: a) The desired power transmission of gears; b) To realize the desired motion, a string can link the two discs; and c) By introducing a second string, the gears can transmit power in both directions.

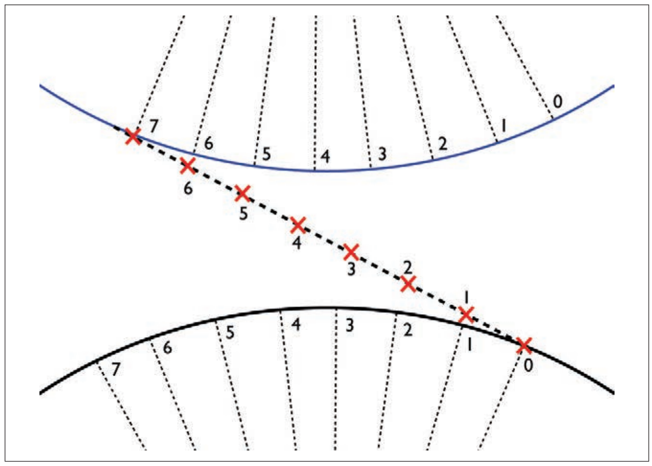



**FIGURE 10.** A close-up of **Figure 9b**, with markings showing the position of the string and angles of each disc as power is transmitted.

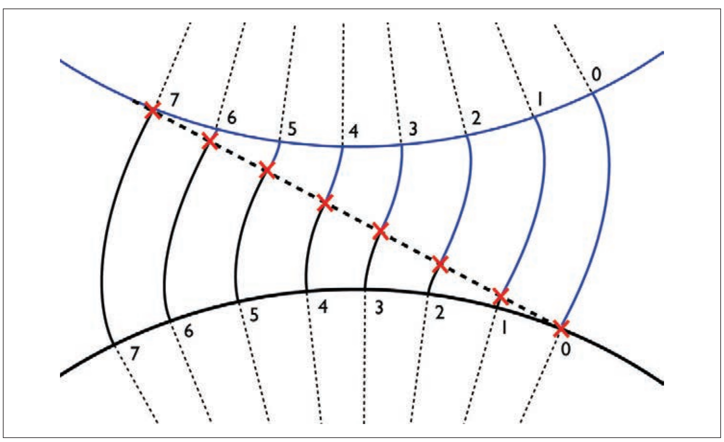



**FIGURE 11.** Using the markings in **Figure 10**, an involute shape is built incrementally.

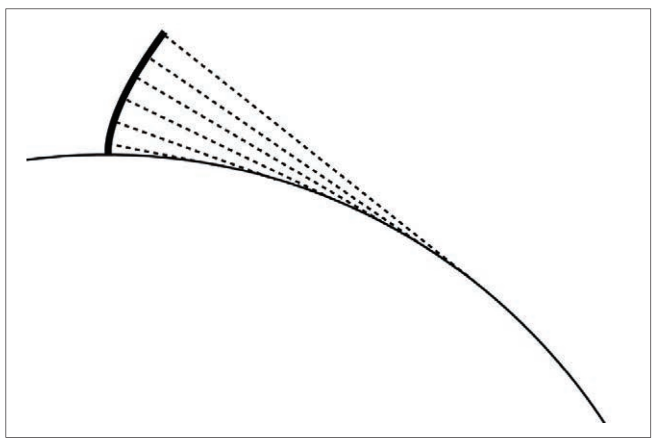


**FIGURE 12.** The involute shape as created from unravelling a string from a circle.

derivative gain, it is clear that the overshoot is severely dropped and the output position settles much more quickly. The derivative gain acts to slow down the motor if the motor is moving fast, and once the position becomes close to the desired location. The motor was set to 50 degrees, but notice that both of the controllers still have a little bit of error at the end of the test.

**Figure 8** shows the second test, comparing the P and PID controller. Here, the P gain was set very low. In the P test, the controller struggled to get the output position error less than three degrees. The PID controller had the integral term which integrated the error over time. As a result, the output angle was tightened up right to 50 degrees.

The PID and PD controllers are the most commonly used controllers, especially for servo control. The controllers work great for any mass-spring-damper system, but not all systems can be easily modeled in this form. The complexities of the hexapod cannot make perfect use of the PID controller because of the geometries involved, but the adjustable gains can be set whenever the lag is either free or in contact with the ground. Fortunately, this flexibility can help to tackle some of the tougher optimal control strategies.

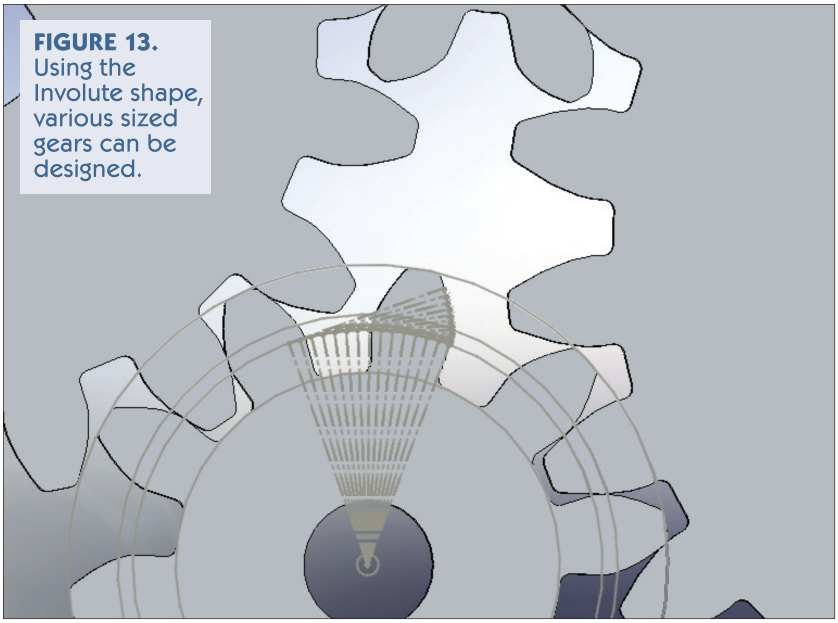


**FIGURE 13.** Using the Involute shape, various sized gears can be designed.

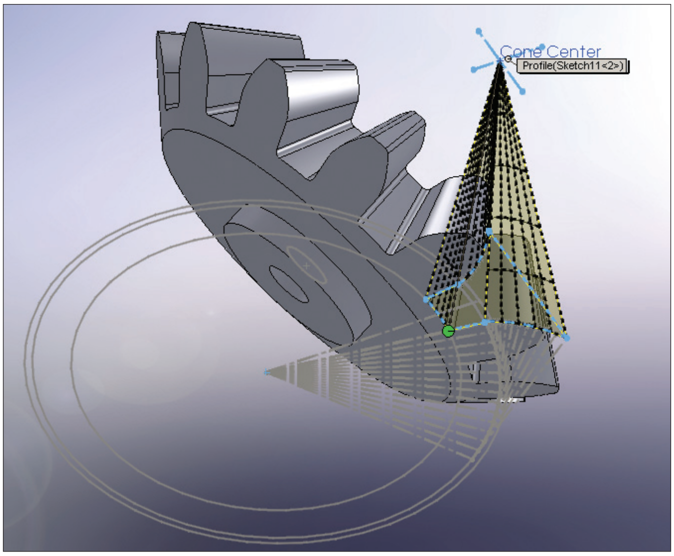



**FIGURE 14.** Again using the Involute shape, bevel gears are created using a lofted cut.

## Gears

The 3DOF head mechanism in the hexapod is rather intricate, involving many spur and bevel gears. Not knowing proper gear design at the time, I simply made some arbitrary triangular cuts in a disc and meshed them in Solidworks the best that I could get them to fit. This resulted in improper motion and teeth colliding, and a significant breaking-in period was necessary to make it usable. If you look at a professionally made gear, you will notice that the teeth seem to have a small flat part, then curve up.

Recently, I looked into designing proper gears using the involute curve method. Originally, I did not know why the curve was important, but after some research the curve finally made sense. I'll attempt to explain why it makes sense to use.

## Gear Purpose

Obviously, a gear is used to transfer circular motion to another circular device. **Figure 9a** shows the general motion. If we were to simply use circular discs, then they would need ideal friction to fully transmit the power. Unfortunately, ideal friction does not exist (it would cause some wear if it did exist). Another method to transfer this motion is to instead put a piece of string wrapped around each disc. **Figure 9b** shows separating the discs and placing a piece of string to transmit power. A nice aspect of using the string is that the transfer of motion between each disc is linear. This will transfer the power well, but only allows a single direction of power transmission. A setup like the one in **Figure 9c** could work for both directions, but the turning range is limited, strings may rub, etc.

Converting the circular motion to linear motion seemed to work nicely, so the question is whether a tooth shape can still transmit the motion linearly. **Figure 10** shows a close-up view of the string between two discs. As the angle of the top disc moves from 0 to 7, the string moves from positions 0 to 7, causing the bottom disc to move from 0 to 7. Let's look at the bottom circle. Ideally, if we were to replace the string with a shape for the bottom disc, the shape surface would be perpendicular at the string, thus transmitting power in the same manner as the spring.

Looking at **Figure 11**, the shape is incrementally built for the bottom circle going from points 0 to 7. So, at point 0, the shape is nonexistent. Moving to point 1, the shape begins to form. At point 2, the shape is built up a little

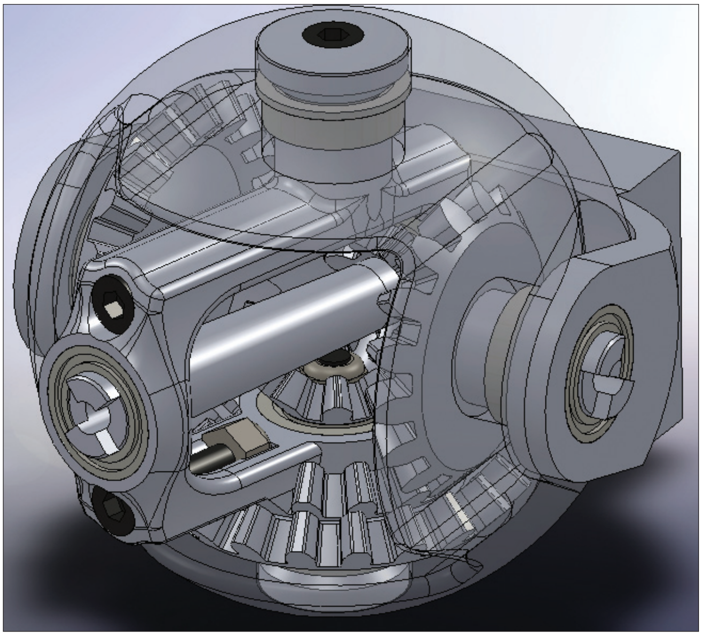


**FIGURE 15.** The newer head mechanism with properly designed bevel gears to enable 3DOF for a camera mounted at the front output shaft.

more. This continues all the way for the entire length of the line. This is similarly done for the top disc, and since the discs are of the same radius they generate an identical curve. The other way to think about the shape is by unwinding the string from the disc. This unwinding is shown in **Figure 12**.

Using the shape, the full tooth can be made. **Figure 13** shows a set of spur gears made using this method with various numbers of teeth and therefore, different diameters.

Also using this shape, a slightly more complex cut can be implemented to design bevel gears as in **Figure 14**. With the use of bevel and spur gears, I am now designing a proper head mechanism which will be smoother, and may not require any break-in time. **Figure 15** shows the usage of various bevel gears in a differential type mechanism to get 3DOF with a camera mounted on the front output shaft.

When designing these gears, I followed a large part of the tutorial on the **cartertools.com** website. Please see the **Links** sidebar to get a full understanding of the gear design process.

## Next Month

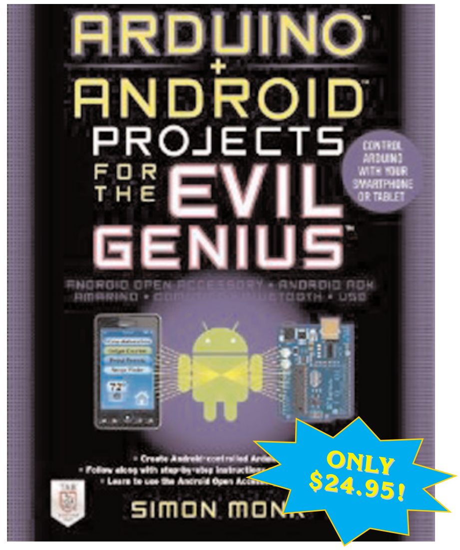
Now with properly designed gears, I can begin working on building parts out of aluminum. I will show some of the complications of going to aluminum and discuss part of the leg redesign needed to assemble all the components. I will also be getting new computer hardware to make the system more capable of advanced behavior. Finally, I will be assembling the complete system and will discuss my thoughts on the components as a whole. **SV**

# The *SERVO* Webstore

<u>*Attention*</u> Subscribers ask about your discount on prices marked with an *

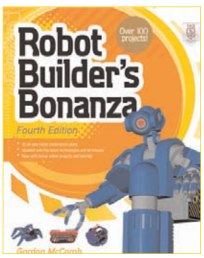
## CD-ROM SPECIALS


## NEW RELEASE



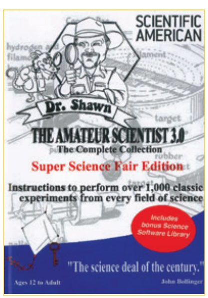

## ROBOTICS

### Robot Builder's Bonanza, Fourth Edition
by Gordon McComb



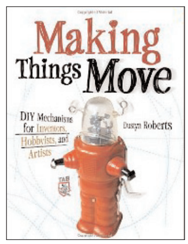
*Robot Builder's Bonanza, Fourth Edition* includes step-by-step plans for a number of motorized platforms. The book is described as a compendium of robotics topics, containing more than 100 projects, including 10 robot designs new to the fourth edition. These modular robots are low cost, and are made to be reproduced by readers with no training in mechanical construction.
**$29.95***

### The Amateur Scientist 4.0 The Complete Collection
by Bright Science, LLC



There are 1,000 projects on this **CD**, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!
**$26.95**

### Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists
by Dustyn Roberts



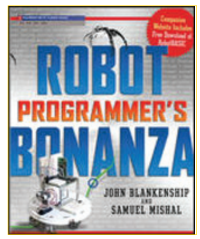
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.
**$29.95***

### Build Your Own Humanoid Robots
by Karl Williams

GREAT 'DROIDS, INDEED!

This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.
**$24.95***

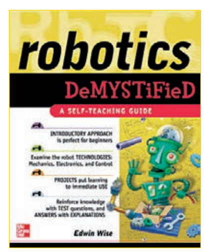
### Robot Programmer's Bonanza
by John Blankenship, Samuel Mishal



**The first hands-on programming guide for today's robot hobbyist!**
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.
**$29.95**

### Robotics Demystified
by Edwin Wise

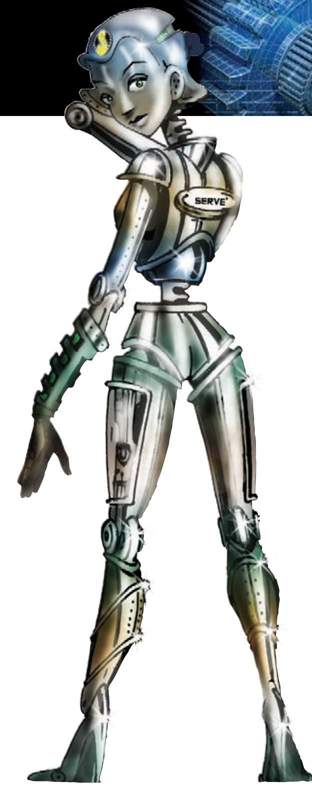
YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

**We accept VISA, MC, AMEX, and DISCOVER**
**Prices do not include shipping and may be subject to change.**

# To order call 1-800-783-4624




## *SERVO Magazine* Bundles


Published by T & L Publications, Inc.

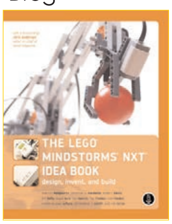
**$57** per bundle

**Save $10 off the normal price!!**

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

## LEGO MINDSTORMS NXT Idea Book

by the Contributors to The NXT Step Blog

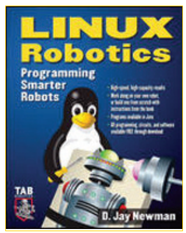



If you're serious about having fun with LEGO® robotics, you've come to the right place. The team behind The NXT STEP blog — the authoritative online source for MINDSTORMS® NXT information and advice — has packaged its considerable skills and experience in this book. Inside, you'll find some of the team's best ideas for creating cool and sophisticated models, including instructions for eight robots you can build yourself.

**Reg $29.95 Sale Price $24.95**

## Linux Robotics

by D. Jay Newman

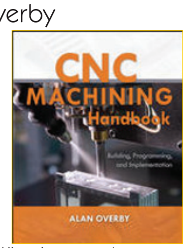



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

**$34.95**

## CNC Machining Handbook: Building, Programming, and Implementation

by Alan Overby

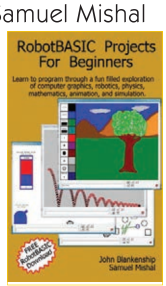



The *CNC Machining Handbook* describes the steps involved in building a CNC machine and successfully implementing it in a real world application. Helpful photos and illustrations are featured throughout. Whether you're a student, hobbyist, or business owner looking to move from a manual manufacturing process to the accuracy and repeatability of what CNC has to offer, you'll benefit from the in-depth information in this comprehensive resource. **$34.95**

## SPECIAL OFFERS

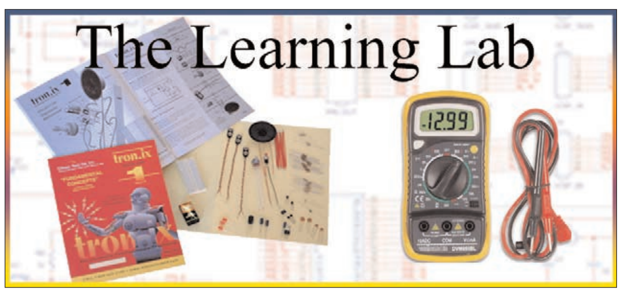
## RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal




If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

**Reg. Price $14.95 Sale Price $9.95**



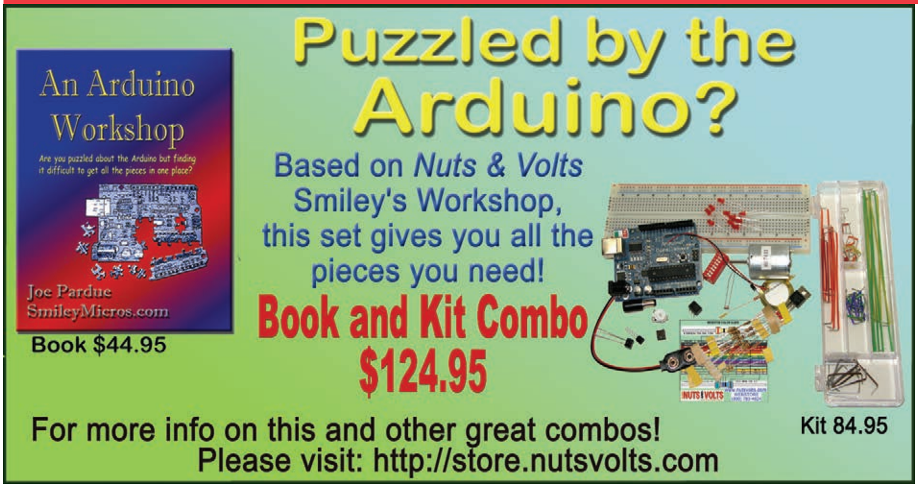

**Technology Education Package for Everyone Starting in Electronics**

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

**Regular Price $79.95** **Subscriber's Price $75.95**

# Or order online www.servomagazine.com

## SPECIAL OFFERS




### Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley

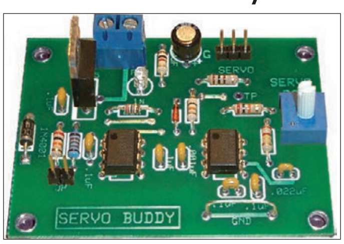
*Forbidden LEGO* introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse.


**Reg $24.95 Sale Price $19.95**



## PROJECTS

### The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.

**For more information, please check out the May 2008 issue or go to the SERVO webstore.**

Includes an article reprint.

Subscriber's Price **$39.55**

Non-Subscriber's Price **$43.95**

### 3D LED Cube Kit

**From the article "Build the 3D LED Matrix Cube" as seen in the August 2011 issue of *Nuts & Volts Magazine*.**

This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.

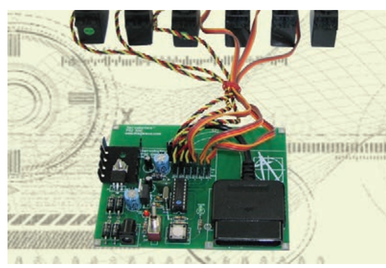
Colors available: Green, Red, Yellow & Blue.

Jig and plastic cases also available.

Subscriber's Price **$57.95**

Non-Subscriber's Price **$59.95**

### PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

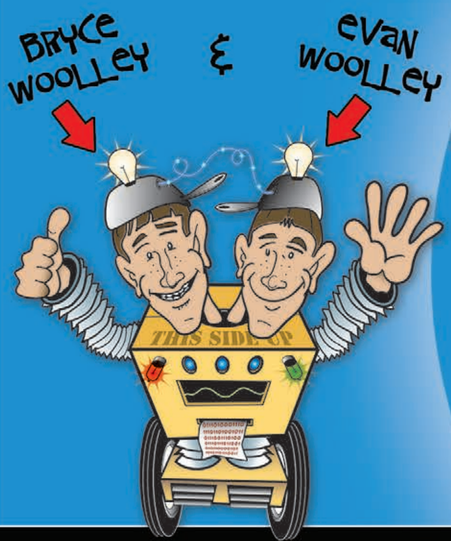
For more information, please see the February 2011 edition of *SERVO Magazine*. Assembled units available!

Subscriber's Price **$79.95**

Non-Subscriber's Price **$84.95**

# Twin Tweaks

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

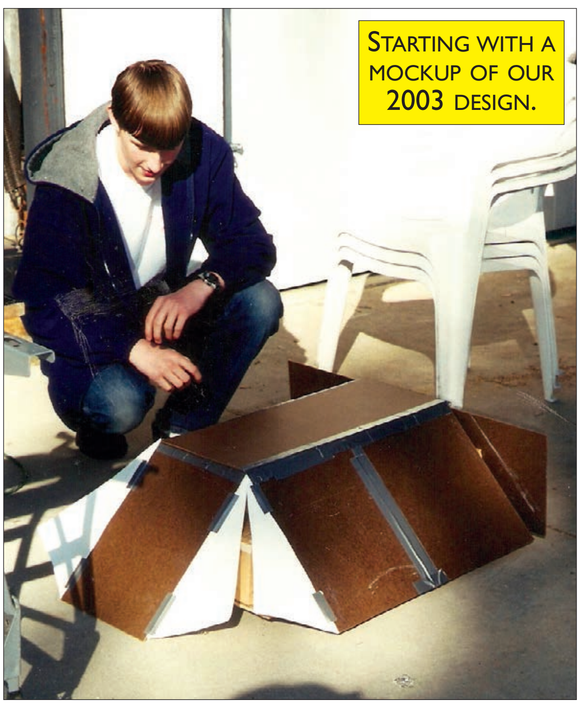



## THIS MONTH: *Looking Backward: 2012-1989*

TEAM 1079, CIRCA 2003.

The beginning of a new year is always an exciting time. Fresh resolutions are still unbroken, the hectic hubbub of the holidays is subsiding, and perhaps most importantly a new season of the FIRST Robotics Competition (FRC) is kicking off. January is the time when teams of intrepid high school students, nay, roboticists, gather for the unveiling of a devilishly clever game that will monopolize their brainpower for the next six weeks. The anticipation of such an unveiling rivals and perhaps even eclipses the delight of descending upon the gifts under the Christmas tree mere weeks earlier.

It doesn't seem like that long ago when in 2003, rookie Team 1079 began churning out ideas in a brain thunderstorm after the game Stack Attack was revealed. Now that it's been a few years since our days as FIRST rookies (and many years since the founding of the FIRST Robotics Competition in 1989), we can really assess what kind of impact it's had on our educational and career paths. We also reached out to some of the folks at LARobotics to get a sense of how the FIRST program has evolved over the years, and what rookie roboticists taking their FIRST steps can expect for the future.

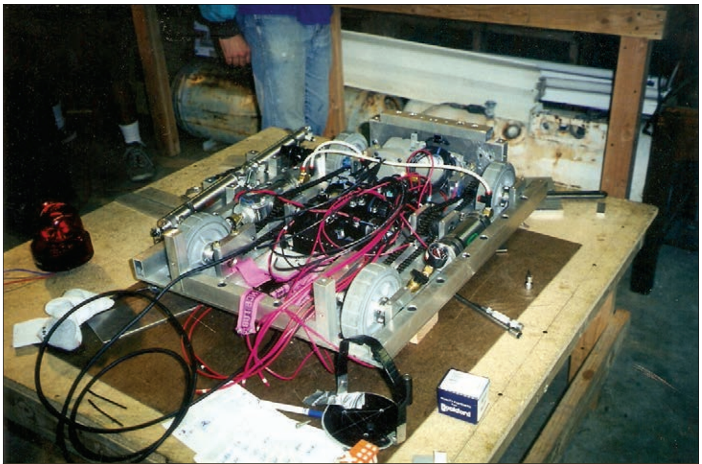


STARTING WITH A MOCKUP OF OUR 2003 DESIGN.

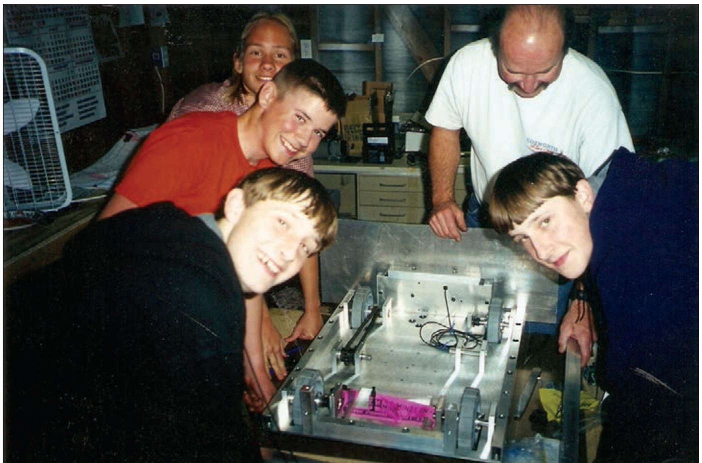
## First FIRST Steps

The first major challenge for any FIRST team is deciding to take the plunge in the first place. A major component of this initial obstacle is a crisis of confidence of sorts. Robots by their nature are pretty intimidating. The popular image of robots as the urbane C-3PO or the deadly Terminator have perpetuated the idea that robots are inevitably extremely complicated; something reserved for PhDs, mad scientists, or the near future when we'll also be consuming all of our meals in pill form as we fly around on jetpacks. Granted, things like Roomba and LEGO Mindstorms have helped to dispel that illusion a little bit, but for many, robotics is an enigmatic and daunting field.

This obstacle affects established teams and prospective teams alike. For established teams, potential new team members can see the success of their classmates, but the lingering doubts about "Yeah that's cool, but could *I* do that?" still remain. This was definitely a challenge for Team 1079 in later years. For completely new teams, this difficulty is even more pronounced. Prospective team members don't have the robots of



Building a robot can be a daunting task ...



But there's nothing more inspiring!

yester-season to act as an encouraging proof of concept.

Fortunately, one of the tools that we used with Team 1079 has evolved into something bigger and better over the last few years, and should be the perfect way for old and new teams alike to entice new members into joining. As the founding members of Team 1079 began to graduate and we needed to inspire a new wave of team members, the FIRST VEX Challenge (FVC) was being introduced. The VEX Robotics Design System has been a perennial favorite of ours ever since we first used it in the Science Olympiad competition as covered in the July '05 issue of *SERVO*.

The VEX starter kit runs about $200, and the FIRST VEX Challenge gave teams a rigorous and structured academic contest that was similar to FRC. The smaller scale of the FIRST VEX Challenge was a great way to convince hesitant students that they too could build robots – the up front costs are lower, the parts seem more familiar, but the thrills of competition are still undeniably present. The 2006-2007 season of the FIRST VEX Challenge featured challenges as difficult as any FRC game – the Hangin-A-Round game challenged teams to negotiate a huge Atlas ball that dwarfed the robots and to hang on a bar in the middle of the field. The Team 1079 contingent for the FIRST VEX Challenge was a mixture of veteran team members entering their senior year, and rookie freshmen and sophomores. And while these students may have gone into the FVC a bit unsure of what kind of contribution they could make, they came out of it inspired and excited to build a full scale FRC robot.

The importance of smaller scale competitions is something that was echoed by Nancy McIntyre, currently the Los Angeles FIRST Senior Mentor and an active member of the Southern California Regional Robotics Forum (SCRFF) during Team 1079's heyday. Nancy also observes that another important way new teams can be inspired to take the plunge into FIRST is by mentorship from an established team. We remember when Team 1079 was first starting, we were taken under the wing of Team 812 from The Preuss School in San Diego. The Midnight Mechanics provided invaluable assistance regarding the intricacies of fundraising and the techniques of building a team brand. Even from a distance of about 60 miles, Team 812 was definitely instrumental in giving Team 1079 the tools and confidence to take the FIRST plunge. And with the proliferation of FIRST teams, prospective newcomers should always be able to find a nearby mentor, especially with folks like Nancy and LARobotics acting as matchmaker.

## Starting Small

The most powerful thing a mentor can share with a new team is simply the knowledge gained from experience. A good example of that might be knowledge about the smaller scale events mentioned previously in case a new team is looking for a more manageable challenge to help them get their feet wet. A mentor who is on top of all of the changes and improvements in the FIRST program is critical, because the smaller scale competitions available as part of the FIRST family of programs have evolved significantly since Team 1079's golden years. After a few years, the FIRST VEX Challenge became the FIRST Tech Challenge (FTC), and the VEX Robotics Design System was replaced by a modified LEGO Mindstorms kit. Fans of VEX need not despair, however, because the VEX Robotics Competition lives on. LARobotics spotlights and supports events for both the FIRST Tech Challenge and the VEX Robotics Competition, and the existence of several ways for prospective teams and competitors to get involved can only be a good thing.

The FIRST Tech Challenge is based on a modified version of the LEGO Mindstorms kit. The NXT brain is used, but new structural components — called TETRIX — are available. TETRIX parts are more substantial than their LEGO counterparts because they are made from aluminum instead of plastic, and robots for the FTC are motorized by TETRIX motors and servos. In many ways, the FTC is simply a meticulously scaled down version of the FRC. Teams are given a kit of parts and a limited time to build and test their robots. The games are competitive and task-oriented, much

like the large scale contests of the FRC.

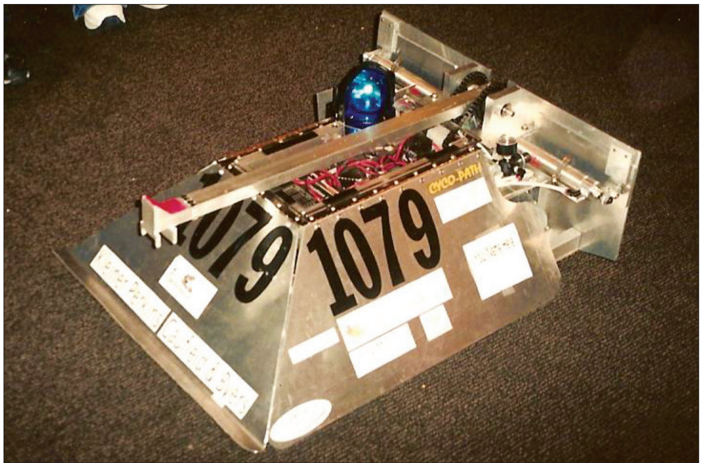
The game for the current season is Bowled Over, where teams have to manipulate objects like racquetballs and ball crates. The matches generally start with an autonomous mode, just like the FRC. The autonomous mode was one of our favorite aspects of the FIRST game. It gave us a chance to practice programming skills, but the consequences aren't catastrophic if your program doesn't work out – the few seconds at the beginning can be used to score a big advantage, but it's hardly determinative. And with the FTC, teams get to practice programming skills with accessible NXT-G or industry favorite LabVIEW.

That is what we think is one of the greatest aspects of the smaller scale competitions offered by FIRST – the FTC scales down the size of the robots to scale down costs and intimidation, but the real world applicability of the skills learned in the competition are on par with those gleaned from the FRC.

In a sense, starting smaller is what helped Team 1079 get going too. Our first robotics endeavor was with combat robots and Botbash, and while Troublemaker and Twibill Trouble were still fairly sophisticated robots, many of the parts were donated from Cosworth and registration fees were pretty low. When the group of students that would eventually become Team 1079 first came together, we were designing another combat robot. That combat robot was never actually finished because we jumped into our rookie year of FIRST before it could come to fruition. The reason we were able to get a group of high school freshmen and sophomores to commit to building a huge and complicated robot in just six weeks is because the smaller scale project had gotten them excited about problem solving and had given everyone the impression that they too could build a robot. That said, the object manipulation aspects of the game and the six-week build time were a bit of a shock to the system – the FIRST game was a bit more nuanced than building a robot to kill or be killed. If we had the opportunity in 2002, we think something like the FTC would have been the perfect challenge for a nascent Team 1079. For teams starting in 2012, opportunities like the FIRST Tech Challenge and the VEX Robotics Competition abound, and they're the perfect way to build experience and confidence in new roboticists.

## Moneybot

Even when you get a team excited about the gargantuan task of building a sophisticated robot in a mere six weeks, practical obstacles abound. Chief among these obstacles is fundraising. We were curious about how teams handled fundraising in this age of belt-tightening and austerity. Team 1079 was lucky enough to get our rookie year funded by venture capital firm Kleiner Perkins Caufield & Byers, and subsequent years were supported by our local economic development counsel and some tenacious sponsors (Crowder Machine & Tool and Flashpoint Machine will always have special places in our hearts).

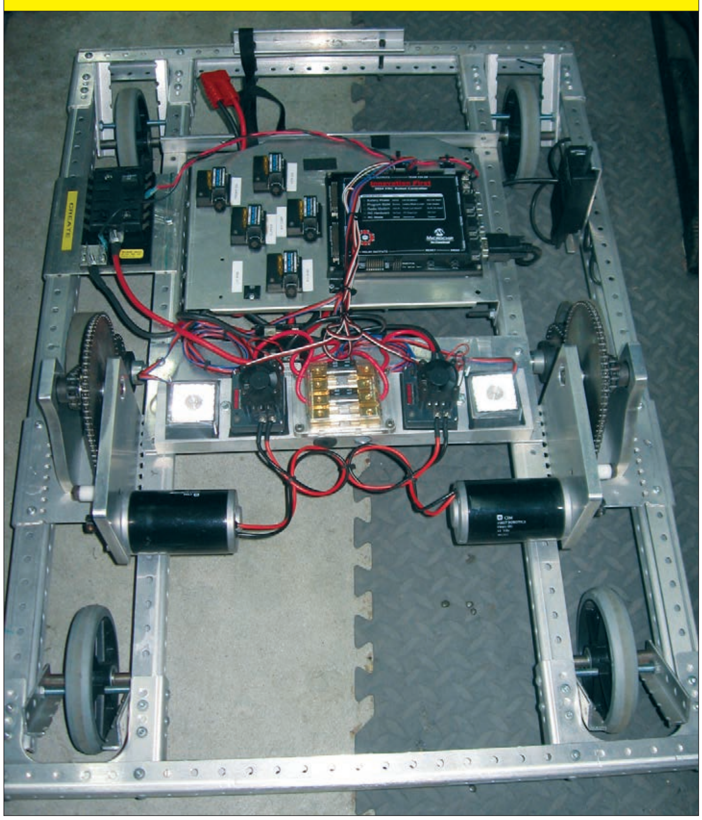


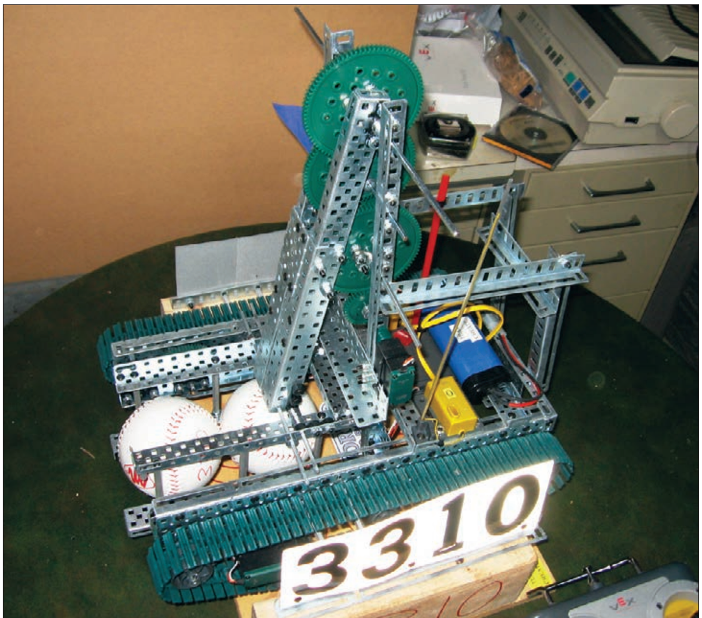

MO, OUR BELOVED ROOKIE YEAR INCARNATION OF MASSIVE OVERKILL.

Thankfully, even in tough economic times there are still plenty of companies that recognize that FIRST is an extremely worthwhile investment. Companies like Boeing and JCPenney provide numerous grants that cover the cost of registration. Another major sponsor that was extremely active during our years in FIRST and still remains an ardent supporter is Raytheon. Raytheon provides support with grants for teams, sponsorship for regional contests, and scholarships for individual students.

The continued involvement of core sponsors like

PROTOBOT, AN OFF-SEASON ROBOT IN SEARCH OF A COMPETITION.





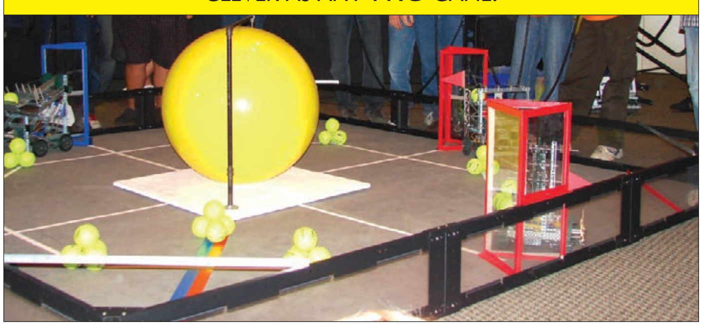
TEAM 1079'S ENTRY INTO THE 2006 FIRST VEX CHALLENGE.

Raytheon really speaks to the enduring benefits that FIRST provides both to students and engineering companies. In addition to funding, Raytheon provides support in the form of mentorship — something that provides students with a valuable window into the work of a real world engineer.

The investment of time given by professional mentors is one of the things that really made an impact on us during our time with FIRST. One of our favorite events sponsored by SCRFF was the Fall Workshop at Cal State Northridge, where professional mentors from companies like Raytheon and Northrop Grumman would take time on a Saturday to teach classes to FIRSTers on topics ranging from 3D drafting to programming, to arms and lifts. It was a veritable merit badge day of robotics, with students filling notebooks with the words of wisdom given by mentors as excited to be there as the students were. Even though the FIRST program has evolved and even though it has been years since our yearly treks to CSUN, the Fall Workshops are still going strong.

Even with all that support, though, the FRC is still a major investment of money and time on behalf of the teams and their mentors. Once again, the introduction of smaller scale competitions like FTC has proven to be a lifeline for smaller and newer teams. FRC can be pretty capital intensive – the kit of parts, the software, and the shipping doesn't come cheap (rookie registration runs about $6,500), especially if teams are the first FIRST team in an area because local businesses may be unsure about supporting a program that they don't know much about. The registration fee and kit costs for FTC come to about $1,200, and the VEX Robotics Design System starter kit is only about $200.

THE FIELD FOR HANGIN-A-ROUND — JUST AS CLEVER AS ANY FRC GAME.



Once teams have an FTC or VRC season under their belts, raising funds for the FRC is typically much easier. Teams have the confidence of a successful season, a physical robot they can demonstrate, and perhaps even an award or two. One of the most powerful marketing tools for Team 1079 for wooing sponsors was to demonstrate our rookie season robot MO. MO became quite the well-traveled bot, making demonstrations everywhere from JPL to a meeting of our local Economic Development Counsel.

We would always bring some totes with us, because a demonstration of the stacking required for the 2003 game Stack Attack is what convinced sponsors that our ragtag group of high school students was quite capable of building a sophisticated robot, and that their money and support would result in something tangible and impressive. The same holds true for veterans of the FTC and VRC that want to raise the money necessary to get into the big leagues of the FRC. Even though the robots are smaller, the ability to manipulate the objects of a devilishly clever FTC game is an impressive feat, and the inspiration instilled in the students is undeniably evident to prospective sponsors.

## Uncurb Your Enthusiasm

Keeping team members excited and inspired during the off-season was always a tough challenge for Team 1079. After the competition, students have to focus on class work that hopefully wasn't too neglected during the frenzy of the build, plus during the summers, people tended to part ways for internships and vacations. One of the ways in which we tried to combat this off-season ennui was by building Protobot – an amalgamation of unused kit parts and a brain transplant from Modos (our second season robot).

Protobot was a great project, and it still serves as a great platform for experimentation (see the May '09 issue for another brain transplant involving Protobot), but in some sense the project was lacking. Protobot was a lot of fun to build, and we were motivated by a recently completed FIRST season to test out some new ideas. We did indeed test out a new drive train design, but the core issue with the Protobot project is that it was somewhat anticlimactic. Once we finished, we were able to drive the robot around the driveway and take it down to the community basketball courts for a more wide ranging test. However, without something more the initial enthusiasm of

the build faded and we didn't get around to testing as many ideas as we had imagined.

We think the shortcomings of Protobot stem in large part from the fact that the project was unstructured – we had just come off of a super-structured FIRST season, but with Protobot we had no game to build for, no teams to compete against. Once again, the smaller scale competitions like the VRC and FTC come to the rescue. The timing of events like the FTC and the VRC is conducive to promoting year-round enthusiasm in your robotics team.

The FTC kicks off in September and concludes with a championship in December. The VRC happens in the spring, with the championship in April. The same qualities that make these events appealing to teams strapped for cash or confidence also make them an excellent way to maintain enthusiasm in the FRC off-season. The FIRST Robotics Competition is a pretty exhausting endeavor – Team 1079 had its share of all-nighters, especially as the merciless shipping deadline drew near.

MO SHOWING OFF AT THE JPL OPEN HOUSE.

Even though every season and every regional left us brimming with ideas for next year, we didn't have the stamina to jump into building another robot that outweighed some of our team members. Even with a small team, though, we were able to compete in the FRC at the beginning of the year and one of the early FIRST VEX Challenges later on. The timing, scale, and structure of the FVC was the perfect way to keep the enthusiasm as high as it was in the thick of the regional competition, and teams today can keep the enthusiasm flowing with both the FIRST Tech Challenge and the VEX Robotics Competition.

## Looking Forward: 2012 — An Even Brighter Future

Being a part of FIRST was certainly the most memorable highlight of our high school days, and several years down the road we can really see the impact that the competition has had on our educational and career paths. The alumni of Team 1079 have gone on to populate engineering schools all across the country, like the Jacobs School at UCSD, the Pratt School at Duke University, and Cal Maritime. This, in turn, has lead to careers ranging from a Naval ship's engineer to patent lawyer. Everywhere Team 1079 went, the practical experience gained by working on a real project was invaluable. Even in engineering school — where students are supposed to be learning the skills to prepare them for real world industry — the practical experience of FIRST was unmatched.

With FIRST, students have a chance to test new ideas and learn skills that range from programming to power tools, all without the school-related stress of grades. FIRST gives students the freedom to experiment, to build things, to fail, and to succeed. The confidence, skills, and values of Gracious Professionalism are not something that can be transmuted by the achievement of an A in an engineering course.

Because FIRST had such an impact on all of us, we are thrilled to see the program growing, evolving, and getting even better than ever before. Not all of these changes, however, have been without speed bumps. One of the major changes experienced by FIRST is a growing pain of sorts – teams are becoming so numerous that some regionals are becoming unruly and cumbersome. Too much enthusiasm sounds like a good problem to have, but nobody likes a logistical nightmare. This concern was simply an opportunity in disguise, and the proliferation of teams has, in turn, led to a proliferation of localized contests.

During Team 1079's tenure, we remember the excitement of inaugurating the San Diego Regional. There was such a sense of community and accomplishment, and the contest was a model of Gracious Professionalism. As local contests proliferate, so will the sense of community that FIRST provides, and so will opportunities for the teams that have yet to jump into the hardest fun students can ever have.

One thing we really want to emphasize to all of those new roboticists taking their FIRST steps this year – FIRST is an experience that stays with you for the rest of your life. It can have a huge impact on your educational and career goals. Team 1079 is proof of what Dean Kamen observes – as a culture, you get what you celebrate. And we're proud to celebrate FIRST every chance we get. **SV**

### RECOMMENDED WEBSITES

**www.larobotics.org**
**www.usfirst.org**
**www.vexrobotics.com/competition**

### SPECIAL THANKS TO

**Nancy McIntyre**

# The NXT Big Thing #16

## Isotope — Part 2 of 3

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com

By Greg Intermaggio

Oh, herro! Werrcome to The NXT Big Thing! This month, we'll build a custom Bluetooth control to Isotope — our linear-actuated all-terrain NXT robot (shown in **Figure 1**). Once built, we'll start on the programming, and then we'll finish it off next month. If you missed the first installment of this mini-series, grab the September '11 edition of *SERVO Magazine* for full build instructions.

## Building the Controller

We've done Bluetooth remote control before for Eddie 2.0, but Isotope is a bit more complex. It will require a controller that's more advanced. Mainly, we need a way to control both the robot's directional movement (forward, backward, left, right) and the linear actuators which allow Isotope to crawl over difficult terrain.

For directional movement, we'll use the NXT buttons like we did back in The NXT Big Thing #11. For each linear actuator, we'll add a motor to our remote control. On each motor will be a gear, and spinning that gear will adjust Isotope's actuators. With that in mind, let's rock 'n roll!

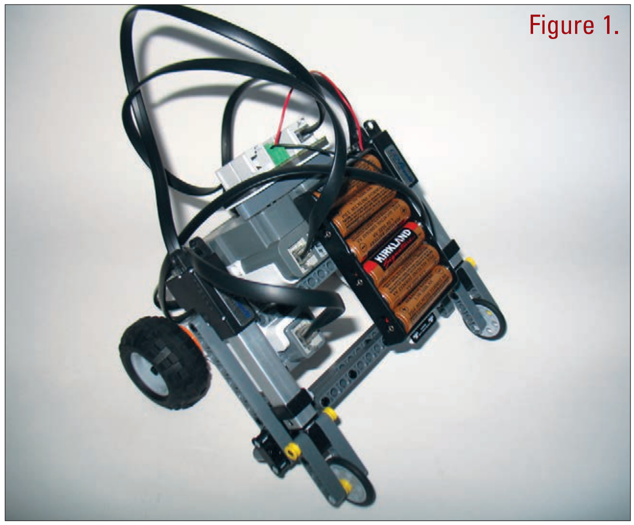
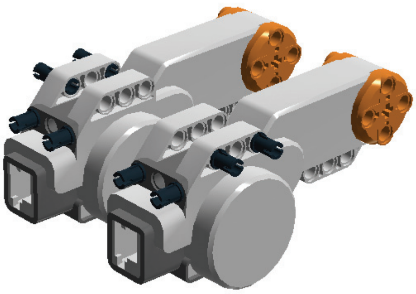
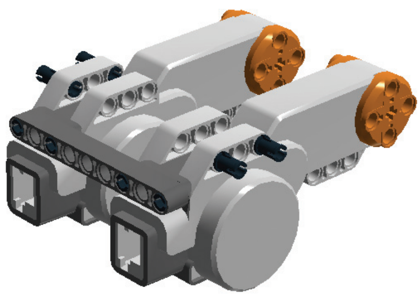


Figure 1.

## Building Instructions: Remote

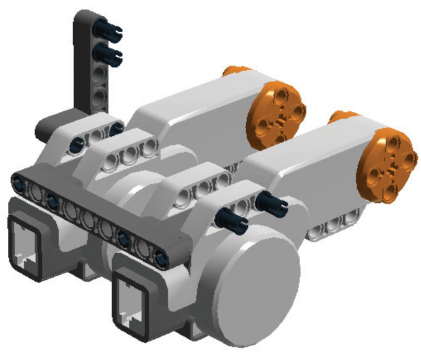
1. Start with two NXT motors, and snap in eight standard friction pins as shown.



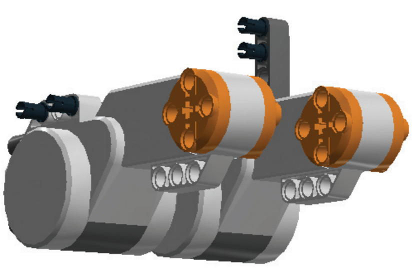


2. Attach a nine-hole studless beam to connect the two motors.

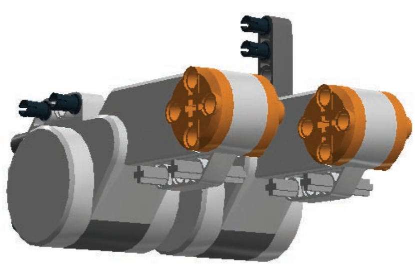
3. Snap on an L-bracket with two standard friction pins.

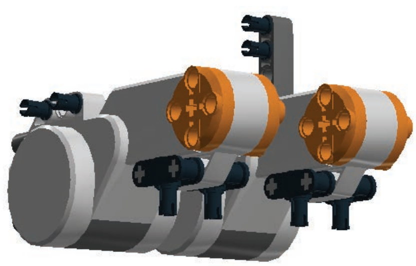


4. Re-orient the assembly so you can see the other side.



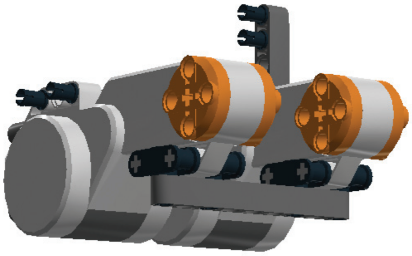
5. Slide in four three-stud axles as shown.



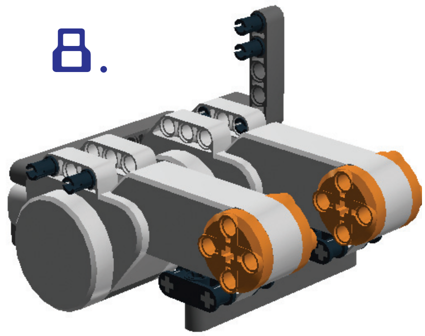


6. Secure the axles, then add four standard friction pins.

7. Attach another nine-hole studless beam to further secure the motor connection.

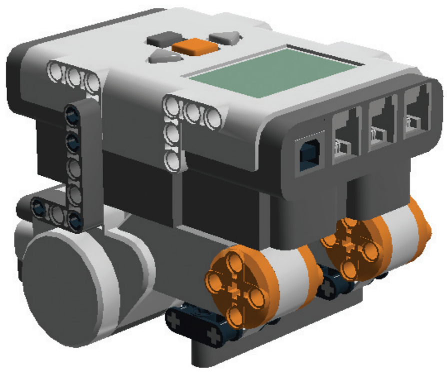


8.

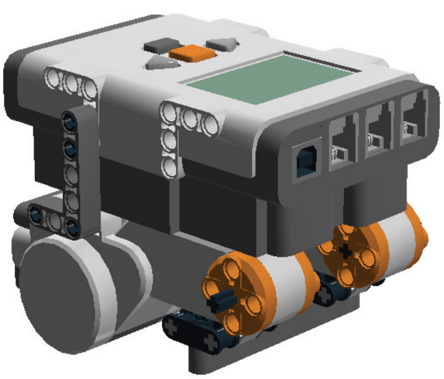


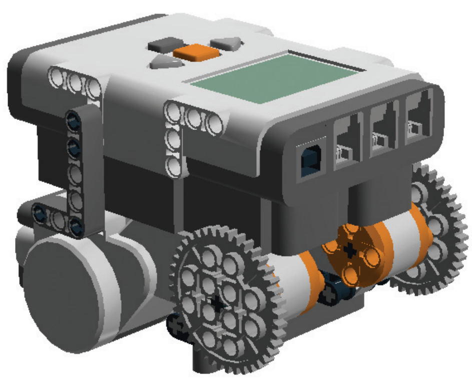
Slightly re-orient the assembly to see the top.

9. Attach the NXT as indicated, then snap in an L-bracket with two standard friction pins to secure the connection.



10. Slide a four-stud axle into each motor, with the end pointing towards the outside of the assembly.





11. Attach 40-spur gears to the axles. Then, just wire the left motor to port B and the right motor to port C, and you're done!

Alrighty! Now it's time to do the initial programming to control Isotope via Bluetooth. If you did our Bluetooth remote in The NXT Big Thing #11, you can re-use the same program. If you didn't, here's a recap.

# Control Test Program Instructions

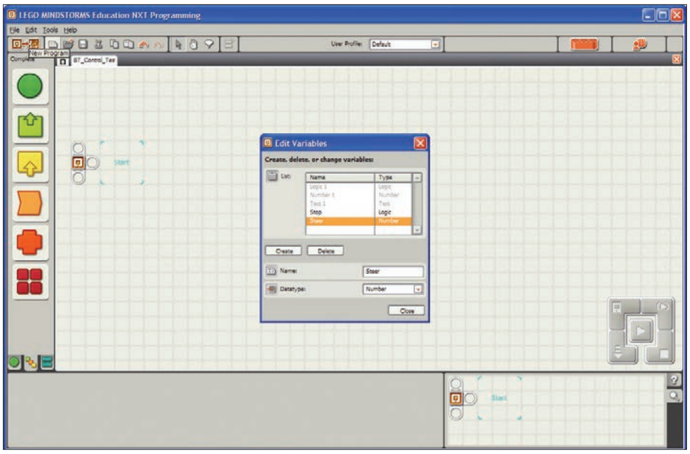


**Figure 1.** Start by creating a new program called BT_Control_Test. Then, click Edit > Manage Variables. Add a logic variable called Stop and a number variable called Steer. Stop will control whether Isotope moves or stops. Steer will control which direction Isotope moves.

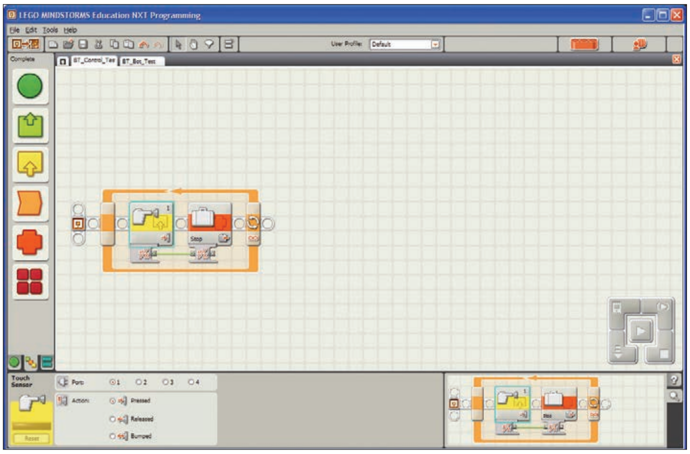


**Figure 2.** Add a loop. Inside the loop, drag a touch sensor block in. Add a variable block and choose the Stop variable. Set the variable to write instead of read. Finally, wire the output of the touch sensor block to the input of the variable block. This means that when the touch sensor is pressed, Stop will be set to true.

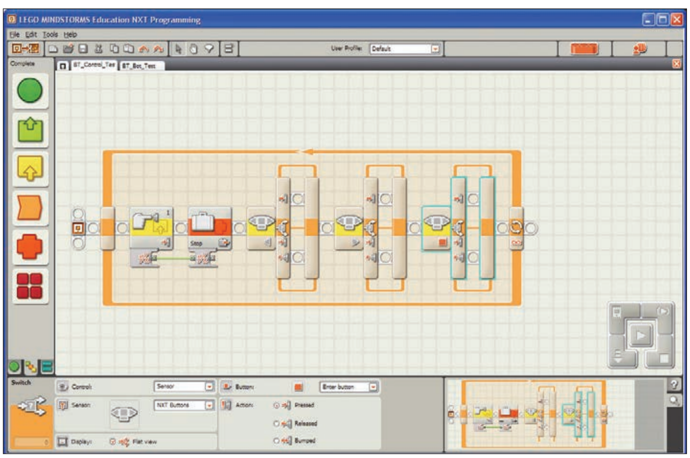


**Figure 3.** Add three switches set to "NXT Buttons" for Sensor. Set the first switch to the left arrow, the second to the right arrow, and the third to the enter button.

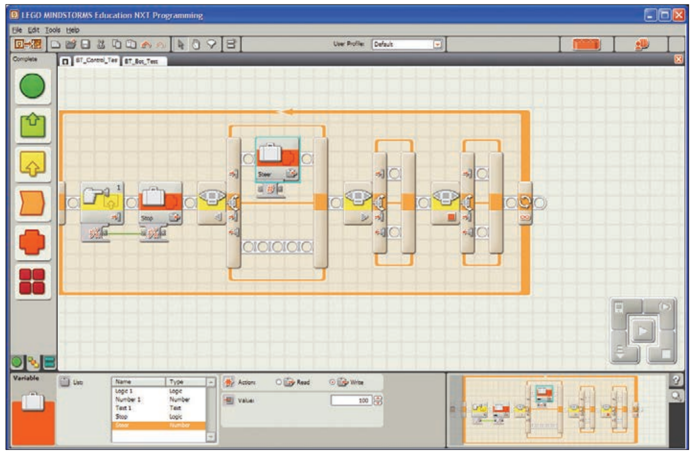


**Figure 4.** Add a variable block to the side of the left arrow switch where the button is pressed. Set that block to the variable Steer, and set it to write. Finally, set the value to 100. Recall that "steering" can be controlled by an integer between -100 and 100: -100 is a full right turn (it's reversed from a full left turn since Isotope has a gear train); 100 is a full left turn. When the left arrow is pressed, we'll ultimately want Isotope to turn left.

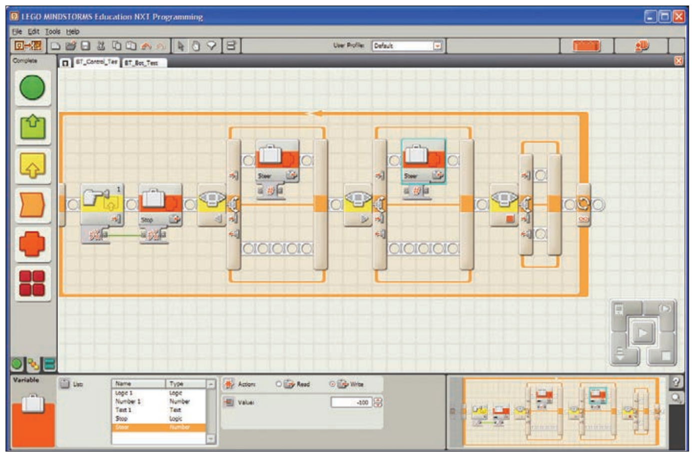


**Figure 5.** Add the same Steer variable block to the right NXT button switch. This time, set it to -100 for a full right turn.

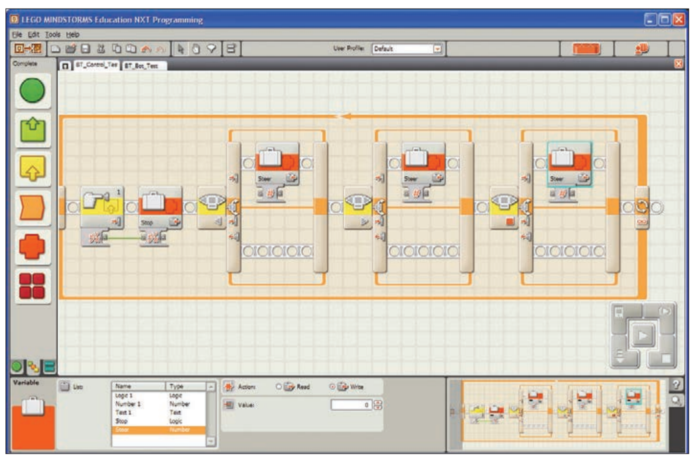


**Figure 6.** Again, add the Steer variable block to the enter button switch. This time, set it to 0 to make Isotope go straight forward when the enter button is pressed.

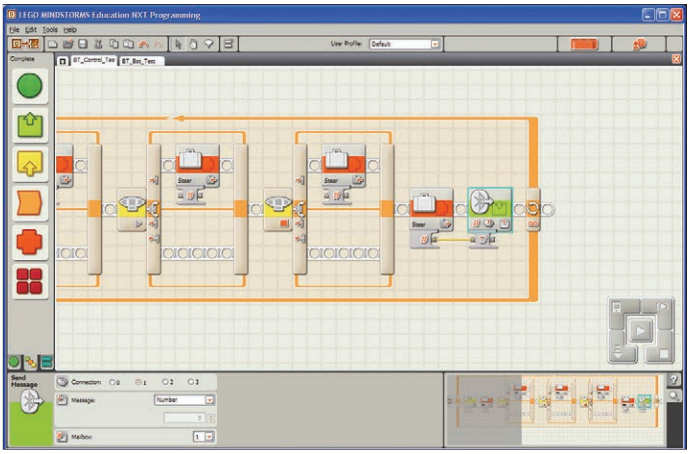


**Figure 7.** Add a variable block to the end of the program and set it to the Steer variable. Next, find and add a Send Message block to the end of the program from the Actions menu. Set the connection number to 1, the message to logic, and the mailbox to 1. Wire the value from the variable block to the number data hub on the Send Message block. You'll have to connect the two NXTs via Bluetooth manually, at which point you choose the connection number. The mailbox number is where the data is sent, and we'll later use it to tell Isotope where to look for those important variables.

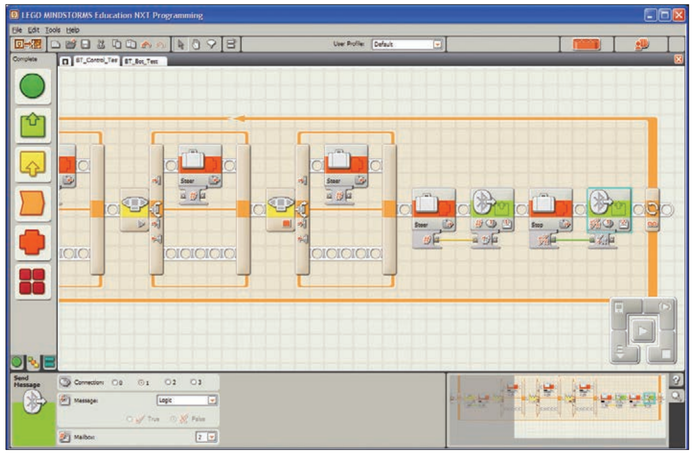


**Figure 8.** Add a variable block and set it to the Stop variable. Add another Send Message block, and this time set it to connection port 1, message "Number," and mailbox 2. This time, connect the value from the variable block to the logic data hub on the Send Message block.

Download the control program to your controller, and the bot program to your robot. Then, connect them via Bluetooth (make sure both NXTs are set to visible so they can find each other!). Once you've conected them via Bluetooth, test your program. If all goes well, Isotope should now respond to your button inputs.

Next month, we'll complete this program by adding actuator control. **SV**

## Bot Test Program Instructions

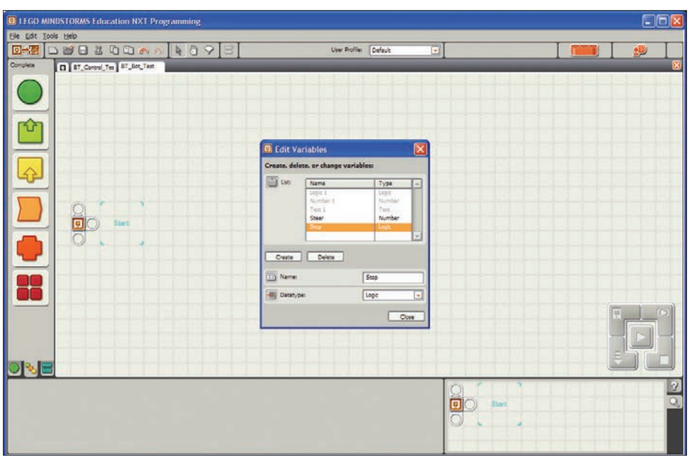


**Figure 1.** Create a new program called BT_Bot_Test. Define a number variable called Steer and a logic variable called Stop. These are the same variables from the control program.

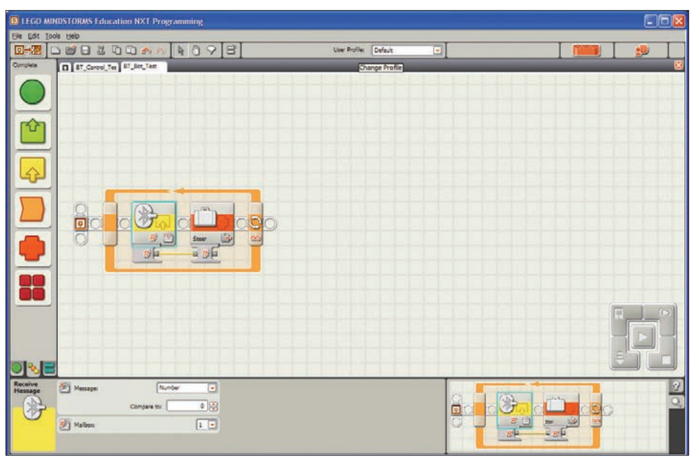


**Figure 2.** Add a loop. Inside the loop, add a Receive Message block from the Sensors tab. Set the mailbox to 1, and set it to the Number Out data hub to write to a Steer variable block.

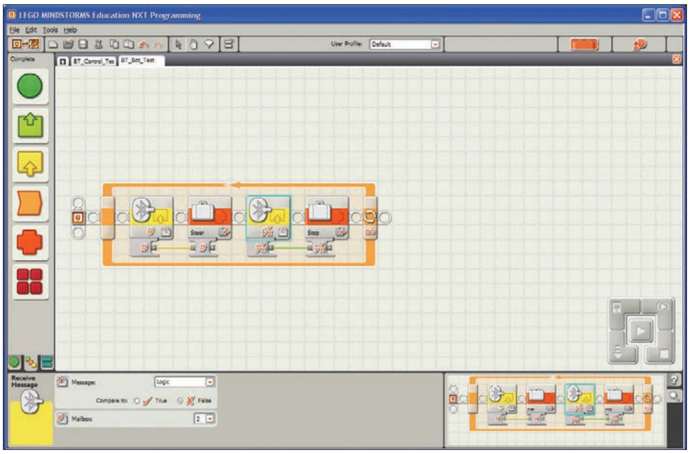


**Figure 3.** Repeat the same process as Step 2, but set the mailbox to 2 and the variable to Stop. Wire the logic out data hub to the Stop variable.

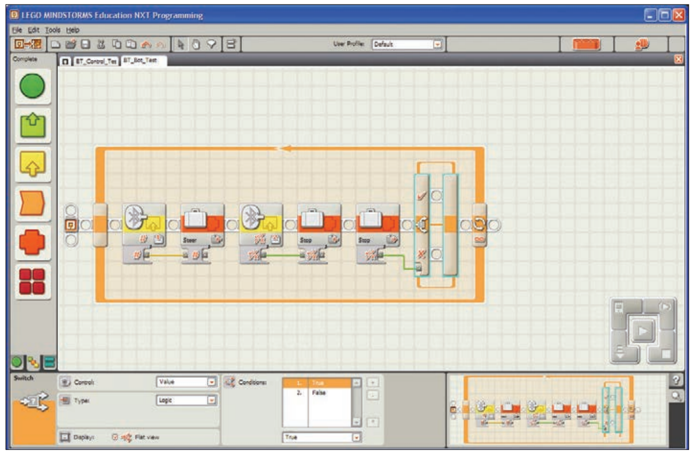


**Figure 4.** Add a Stop variable block set to read, and wire it to a switch set to logic. This will allow us to make Isotope react differently if the touch sensor is being pressed (meaning Stop is set to true), as opposed to if the touch sensor isn't being pressed (meaning Stop is set to false).

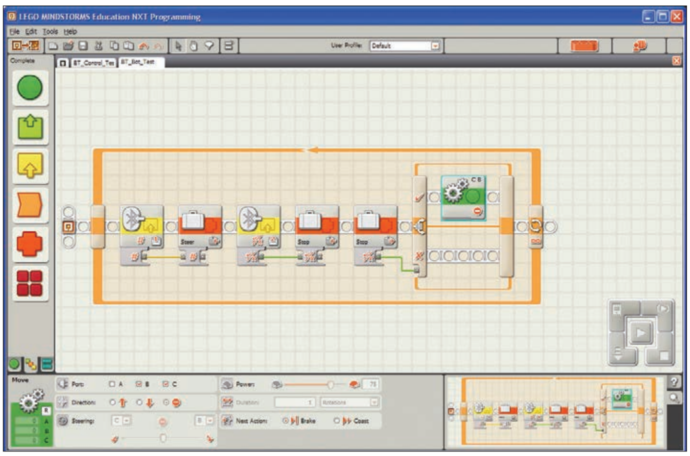


**Figure 5.** If Stop is set to true (meaning the touch sensor is being pressed), we want Isotope to stop moving. Add a Move block set to stop on the true side of the logic switch.

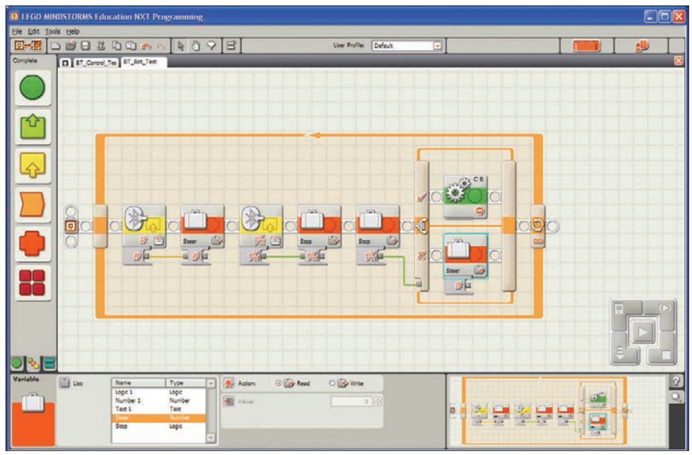


**Figure 6.** Add a Steer variable block to the false side of the logic switch.

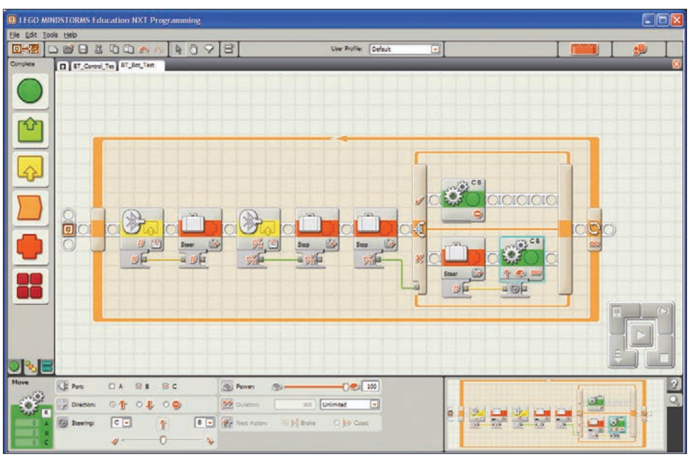


**Figure 7.** Add a Move block and wire the output of the Steer variable to the Steering data hub of the Move block.

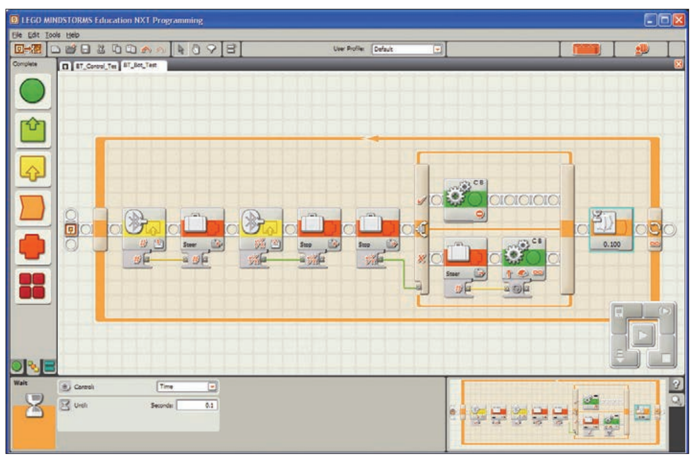


**Figure 8.** Finally, add a Wait for Time block at the very end of the program, setting it for .1 seconds. This will make sure that Isotope doesn't try to look for Bluetooth signals too fast, when they aren't there!

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

# Then and NOW

# TurtleBot: ROS Meets Kinect Meets Create

by Tom Carroll

*Turtle robots have been around since Grey Walter's Tortoise of the late '40s. When hobbyist robots started to be popular in the early '80s, robots such as the Tasman Turtle (shown in **Figure 1**) were popular. They were easy to interface and control by an Apple II or similar computer.*

Newer turtle-type robots such as the BBC Turtle robot shown in **Figure 2** or the Propeller-head Geek Turtle robot shown in **Figure 3** are still popular and are sold at many hobby robot outlets. These robots are simple to construct, interface, program, and operate as they are usually differentially-driven and have a clear plastic shell that shows the interior. They offer beginners a way to learn programming techniques to control a basic robot.

The term 'turtle' has recently taken on a new meaning with the introduction of two turtle-style experimenter's machines that are changing the face of hobby and advanced experimental robotics. I've reviewed many robots over the years, but have rarely come across two entirely different machines at the same time that have really offered so many capabilities and opportunities for serious robotics research. Both machines have a long list of users who have nothing but praises for their robot. With newer, powerful software suites coupled with equally powerful sensors, these two advanced turtle-style robots offer serious experimenters complete robotics packages that would rival the most advanced university level machines of five years ago — but at a fraction of the cost. One is the Willow Garage ROS-based TurtleBot and the other is Eddie from Parallax that is based on Microsoft's Robotics Developer Studio, RDS4. Neither of these two robots actually utilize turtle shell construction and both feature Microsoft's powerful vision sensor: the Kinect. Each robot actually has so many features that I will do a comparison over the next couple columns. I'll concentrate on the TurtleBot this month. Next time, we'll take a look at Eddie.

## The TurtleBot

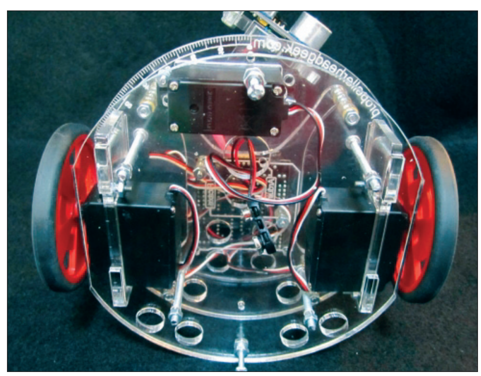
In reading emails and other news items (and talking with WG personnel in early 2011), I heard rumors that WG was developing a small robot to serve as a base for their open source Linux-based software platform, Robot Operating System (or ROS). When I visited their facility in 2010 and got to know their amazing PR2 robot, I was also introduced to ROS which was the software for the PR2. I knew that WG was anxious for the robotics community to discover ROS and to implement it into various robot designs.

Willow Garage was formed in late 2006 by Scott

**FIGURE 1. Tasman Turtle Robot from *theoldrobots.com*.**

**FIGURE 2. BBC Turtle Robot.** ***(Courtesy of Paul Silver at flickr.)***

**FIGURE 3. Propeller-head Geek Turtle robot.**



Hassan as a spawning ground for the development of non-military robotics and open source software. He felt that the impact of innovative ideas in robotics will, in turn, provide opportunities for return on investment of capital. He brought Steve Cousins on board as President and CEO, who helped form WG into a model of innovation incubation.

I have been favorably impressed with Willow Garage since that first visit, and briefly mentioned their new experimenter's platform robot and ROS in last month's column. When the TurtleBot was introduced this past summer, I felt that I just had to look seriously at this advanced experimenter's platform that was significantly cheaper than the PR2. In mid-August, I flew down to Willow Garage in Menlo Park, CA and met with Melonee Wise and Tully Foote — the co-developers of the TurtleBot. TurtleBot is based on the iRobot Create — iRobot's Roomba minus the vacuum cleaner parts — and the Microsoft Kinect — the vision game sensor for the Xbox 360. Having previously seen many photos of the robot, I certainly wasn't expecting anything like the human-sized and very expensive PR2. Though smaller than the PR2, the TurtleBot has capabilities that I've rarely seen on any robot and certainly not one in the range of $1,200 to $1,400 for a 'ready-to-go' machine.

**FIGURE 4. Tully Foote and Melonee Wise with a TurtleBot.** ***Photo by Jimmy Sastra.***

Tully Foote, a Systems Engineer at WG, shown on the left of the TurtleBot in **Figure 4** and Melonee Wise, a Senior Engineer at WG, shown on the right co-wrote the initial proposal at Willow Garage, and then worked on the development of the various hardware components and the TurtleBot-specific software within ROS. It was to be an internal WG project that utilized the just-released Kinect to develop an entry-level mobile robot with many of the capabilities of their PR2. The goal was to produce a mobile robot with sufficient sensor/computing capability to perform multi-robot research and demonstrate how a simple robot design could take advantage of the many features of ROS. They wanted an inexpensive robot that was easy to assemble and was created from basic 'off-the-shelf' components. When visitors to WG saw the robots, they, of course, wanted one, so WG polished the design a bit, and now there are three vendors selling TurtleBots worldwide, with dozens of schools and institutions using them.

All of these systems and components have been available for over a year now and the TurtleBot is about as straightforward as you can get for a basic robot design. There is no cute little turtle shell body and there are no arms or appendages of any sort, however, these and any other device can easily be added at a later time. Experimenters can utilize the basic intelligence instruction set of the Create, add an iRobot Atmel ATMega 168-based command module (shown in **Figure 5**) for the most basic routines, or use the included powerful (for a netbook) Asus 1215N to process the Kinect's vision and other data.

## Key Parts of the TurtleBot

Let's look closer at the four key components of the TurtleBot. The robot uses the Create platform as the base, power source, and for motive power and steering. Kinect serves as the main sensor suite and does so quite well. Willow Garage decided to use the Eee PC 1215N netbook from Asus with 2 GB of RAM and a 250 GB hard drive. This netbook uses the Intel dual-core Atom D525 processor that is powerful enough to handle the demands of 3D data from the Kinect. For graphics, the Asus uses the NVIDIA ION Discrete Graphics Processor. Also included is a power and sensor board with an ADXRS613 single-axis gyro that can measure the robot's yaw rates up to 150 degrees/s. The board also provides regulated 12V power for the Kinect (the USB cannot supply the required current) and is mounted inside the Create. The rest of the components are the four round mounting boards and the spacers for the boards and Kinect mounting. The ROS software is provided on a separate Flash drive and is also downloadable from **ROS.com**.

## Basic Mechanical Construction

The four 12-3/8" mounting plates are perfect for mounting experiments or even an inexpensive robotic arm. The bottom plate is affixed to the Create and has a 5-1/2" by 7-1/2" hole to allow access to the Create's 'cargo hold' and interconnection of various cables. The front of the lower plate is cut off to allow the Create's homing sensor to 'see' the charging beacon. The top plate is cut off at the front so it doesn't shadow the Kinect's field of vision. Originally, I had accidentally mounted the cut-off portion to the rear (to line up the ROS/WG graphics) but found that it affected the Kinect's operation. You can easily do your own modifications such as adding a bungee cord or long twist-tie to hold the netbook in place.

The bottom three plates are separated by two sets of 2" spacers, and the top plate is separated by four 8" spacers. I also tried removing one of the 2" spacers so I could swivel the netbook in and


**FIGURE 5. Create command module installed.**

out for access to the keyboard and screen while it was powered up. The top plate has a series of 1/8" holes spaced 3/4" apart in a 4-1/2" by 7-1/2" pattern that can be used for mounting experiments and sensors. The Kinect is mounted on two 3-1/4" spacers and is set to the back of the robot. It's elevated to allow the Kinect's fixed angle of vision to detect objects as close to the front of the robot as possible. The tilt base is not attached; instead, the body of the Kinect is fastened to the third plate. You have to remove two adhesive-attached strips from the bottom of the Kinect to access the threaded holes. The whole mechanical assembly is very simple and only took me a few minutes.

## The Base: The iRobot Create

There have been literally thousands of robots constructed by using the iRobot Roomba series of robot vacuum cleaners that were hacked to remove only the vacuum system and brushes. iRobot quickly saw the potential of a market for robot experimenters and developed the Create. The basic model is priced at $129. This Create does not include a rechargeable battery but rather a green case that holds a dozen alkaline batteries. I would recommend adding the 3,000 mAh Ni-MH battery and the iRobot fast charger for a total price of $219, as the Create can literally 'eat' non-rechargeable alkaline batteries like they were candy. The $60 iRobot Command module that I mentioned earlier is a great addition if you don't want to use just the ROS software that comes with the TurtleBot. iRobot has what they call the Premium Development Package which includes the above plus a remote control, two virtual walls, and the self-charging base for the charger.

The Create is based on the iRobot Roomba 400 series and is compatible with many of that series' accessories. According to the specifications, it is designed to handle a maximum payload of five pounds and has an extra wheel supplied to stabilize taller robot designs like the robot shown in **Figure 7** — another Create/netbook/camera configuration. The TurtleBot arm shown in **Figure 8** was originally developed by Michael Ferguson. WG summer intern, Helen Oleynikova, worked with the arm and demonstrated it for me. Needless to say, the 14.1 pound TurtleBot weighs a bit more.

The Create's wheel shown in **Figure 9** is beefy. The belt drive from the motor is connected to a planetary gear arrangement, and the shaft encoder (gray and black wires) is used for odometry. With 32 installed sensors (you can add more of your own), a 25-pin expansion port, and a series of 10 nice demos built in the Create is an ideal experimental platform on its own. Using either the command module or your own microcontroller and the Quick Start Guide (which gives you access to the compiler and a series of sample programs using C or C++ or some other programming languages via the iRobot Open Interface) will give you a great start for an advanced robot. If you want to use the Kinect, I highly recommend the use of ROS for its unique abilities with the Kinect sensor's more complex data stream (though Microsoft's RDS4 has been tweaked to also utilize the vast amount of visual data). I would never recommend hacking an old Roomba in an attempt to convert it to a Create-like base, as you will probably end up with a poorly operating machine and not have anything like a Create. Keep the Virtual Walls/Lighthouses though, because they will work with the Create. At $129, you'll be far happier buying a brand new model. Co-founder of iRobot, Helen Greiner, gave me a Create several years ago and I have found it to be the perfect 'already built' base for most robot experimenter's projects. I was quite grateful for the Create because it has so many applications in robot research. It is easy to see why WG chose this capable platform for the TurtleBot.

## Eyes and Ears: The Xbox 360 Kinect

The advent of Microsoft's Xbox 360 Kinect sensor was the true impetus for the development of the TurtleBot and the Parallax/Microsoft Eddie. The Kinect is not just a camera for some robot's processor to make sense of a data stream sent by the camera. It is an intelligent vision/navigation/mapping system that — at a mere $130 — can best systems that cost a hundred times more. Within hours after the Kinect was introduced in November '10, it was hacked, and many photos and hacked data soon appeared on the Internet. By the 60 day point — with sales averaging 133,333 units per day — it had made the Guinness World Records list as the best-selling consumer electronics device of all time.

You can use a Kinect with a Create utilizing a tutorial by Melonee Wise at **www.ros.org/wiki/kinect/Tutorials/Adding%20a%20Kinect%20to%20an%20iRobot%20Create**. This machine was her stepping stone to the TurtleBot. Microsoft has donated 2,500 Kinect sensor systems to FIRST high school competition robotics teams for use during the 2012 season — an ideal next step up for these steadily advancing series of robots. There is so much information available on the Internet about the Kinect itself, Kinect hacking, and robot applications that I won't go any deeper into it here. It is the ROS and the Kinect working together that makes the TurtleBot what it really is.

## ROS: the Soul of TurtleBot

I had some uncertainties about just how adept I might be when installing and using ROS on the Asus netbook in the TurtleBot. As a metal-bender, electron-shover, software and programming have never been my strong suit. Fortunately, Melonee and Tully spent an afternoon guiding me through some of the basics, so I was able to adapt. I did make a few mistakes along the way, however.

After downloading the 99 page tutorial, I sat down and tried to immerse myself in it. Some parts came easily to me and some I really had to think about. I highly recommend all potential buyers of the TurtleBot first sit down with the free downloads of the open source ROS and the variations from **ros.org**, and get used to the way it operates. ROS is far more than a programming language as it is an amazing and quite popular extensible collection of libraries and tools for robotics. One of the nice features is the ability for the community of users and developers to be able to contribute a system package.

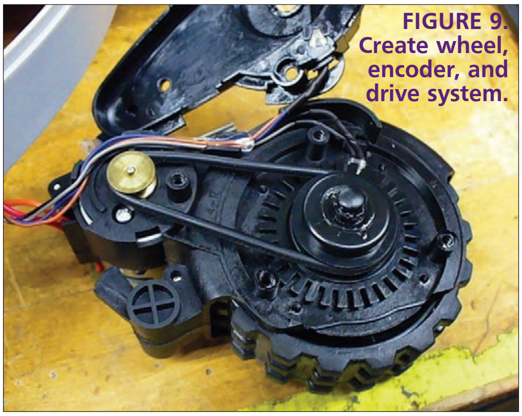


**FIGURE 9. Create wheel, encoder, and drive system.**

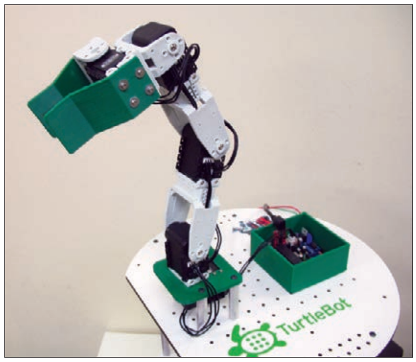



**FIGURE 8. TurtleBot arm with Robotis AX-12 actuators.**

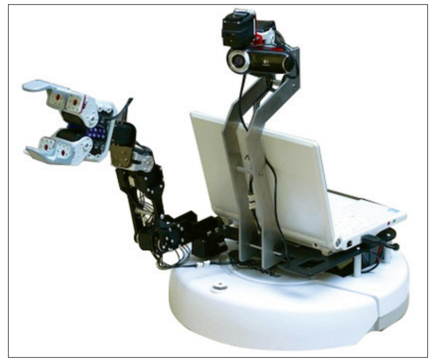


**FIGURE 7. Calliope Development Platform from *robotshop.com*.**

ROS has progressed beyond 'switchyard' — the name given to it at Stanford's SAIL (Stanford Artificial Intelligence Laboratory) in 2007. Further development of ROS is now centered at Willow Garage. It supports Ubuntu Linux and is an entirely open source software that is free for commercial and research use.

After the quick mechanical assembly, the robot is now ready for its software installation. The TurtleBot kit includes a green USB drive that contains ROS and Diamondback (the latest release). At **ROS.org**, you can go to Robots/TurtleBot/diamondback/Robot Setup – ROS Wiki for setup and installation instructions. One thing to remember is to NOT boot up the brand new laptop to bring up Windows because this will make it harder to access the BIOS screen. In fact, you can run into a bit of trouble by calling up Windows Office 2010 to purchase it at a later time — as I did — not thinking about what the netbook was intended for in the first place: the robot's brain. Insert the USB thumb drive into the netbook and turn on the computer while repeatedly depressing "Esc" until the dialog "Please select boot device" appears. Select USB device and hit "Enter." Follow the 'robot setup Wiki' instructions very carefully. You'll shut down, remove the USB drive, re-start, and start up Ubuntu. The next steps are setting up NTP (Network Time Protocol), the root password, and the SDK (software development kit) on the computer. The following steps from the ROS Wiki site are designed to assist you in getting started with the TurtleBot and are a great introduction to understanding ROS:

1. TurtleBot Setup: These instructions will guide you through unpacking your netbook and installing the TurtleBot software.
2. TurtleBot SDK Setup on your computer: These instructions guide you through how to set up your computer to control the TurtleBot.
3. TurtleBot Networking Setup: Establishing communication between the TurtleBot and workstation.
4. TurtleBot Environment Setup: Set up your environment.
5. TurtleBot Bring-up: How to start a TurtleBot.
6. TurtleBot Teleoperation: How to teleoperate your TurtleBot with a keyboard or a joystick.
7. TurtleBot Follower Demo: How to make your TurtleBot follow whatever's in front of it.
8. TurtleBot Odometry and Gyro Calibration: This will show you how to calibrate or test the calibration of a TurtleBot, which is highly recommended when running any navigation-based application.
9. SLAM (simultaneous localization and mapping) Map Building with TurtleBot: How to generate a map using g-mapping.
10. Autonomous Navigation of a Known Map with TurtleBot: This tutorial describes how to use the TurtleBot with a previously known map.
11. Visualizing TurtleBot Kinect Data: This tutorial shows you how to look at data coming from the Kinect camera.
12. Using TurtleBot Interactive Markers: This tutorial describes how to use rviz interactive markers for controlling the TurtleBot.

This article is not intended as a primer on ROS because the subject is so vast. ROS Electric is the next upgrade after Diamondback, and should only be installed after a thorough knowledge of Diamondback, in my opinion. Some people have found learning ROS to be a fairly daunting task. TurtleBot and ROS are really for serious robot experimenters; it is not for a smaller robot running on a microcontroller only. You're going to require a fairly powerful netbook at the least, but once you have your machine assembled and are running the software, you are literally going to be blown away with the numerous advanced capabilities of your machine. If you are even a little familiar with Unix or embedded Linux and know the C++ that ROS is written in, TurtleBot will be ready to go after loading the software. You'll want another computer (Mac or Windows) as a workstation for Wi-Fi teleoperation, preferably with a joystick. If you have some trouble or questions about TurtleBot, ROS, Kinect for Windows SDK, applications, SLAM and gmapping, or any other topic, the TurtleBot wiki pages at **ros.org** are a great resource.

## Final Thoughts

The TurtleBot may look like a simple robot, but the three main components working in conjunction with each other and ROS deliver a machine that is capable of tasks limited only by your imagination. Of course, you can build your own machine by buying or substituting some of the components, but the nice thing about purchasing the full TurtleBot kit is the included software which is already set up for the Create's base and the positioning of the Kinect. After the software is loaded, it's ready to go. **SV**

# ROBO-LINKS


## Advertiser Index

SERVOCITY
A DIVISION OF ROBOTZONE, LLC.
WE HAVE THE PARTS FOR YOUR IDEAS!
Check out our HUGE SELECTION of
GEARMOTORS
Brand New!
6V Micro Gearmotors
Just 0.6 ounces!
Micro-sized!
only $14.99
Lightweight Set Screw Hub
- Incredibly powerful
- All metal gears
- Extremely versatile and durable
- 3mm "D" shaft
- DC Reversible motor
- Choose from 9 different speeds (ranging from 100 RPM - 6,000 RPM)
3mm Gearmotor Pinion Gear
Offset Gearmotors
-Bronze sleeved bearings
-5/16" diameter shaft
-12VDC Operation
-Zinc die-cast case
-DC reversible motor
-Steel gears (Dupont® & Delrin®)
Precision Gearmotors
Ultra Quiet!
Works great with our 6mm bore pinion pulleys
-12VDC Operation
-DC reversible motor
-Solder type terminals
-Heavy duty construction
3-12V Gearmotors
-Heavy duty construction
-DC reversible motor
-Oil bearing design
-Solder type terminals
6mm bore clamping hub
17 speeds to choose from!
Gear attached to clamping hub
Robotzone Gearmotors
-540 type motor
-Hardened steel gears
-1/2" diameter output shaft
-Heavy duty and versatile
CHECK OUT SERVOCITY ON THE GO!
and MOTOR ACCESSORIES
Pulleys & Belts
Super Duty Belt Drive Pan System
Gears & Sprockets
Endcaps & Mounts
Hubs & Adaptors
Like us on Facebook
To view our entire selection of products visit us online at
WWW.SERVOCITY.COM
Specializing in Servos, Radios, Motors, Pan & Tilt Systems,
Batteries, Wiring, Connectors, Gearboxes and more!
Check out our videos on YouTube
www.youtube.com/user/ServoCity
Copyright 1999-2011 Robotzone, LLC. - All Rights Reserved
ServoCity is a registered trademark of Robotzone, LLC.
Phone: (620) 221-0123
E-mail: Sales@ServoCity.com
Prices and availability subject to change without notice.

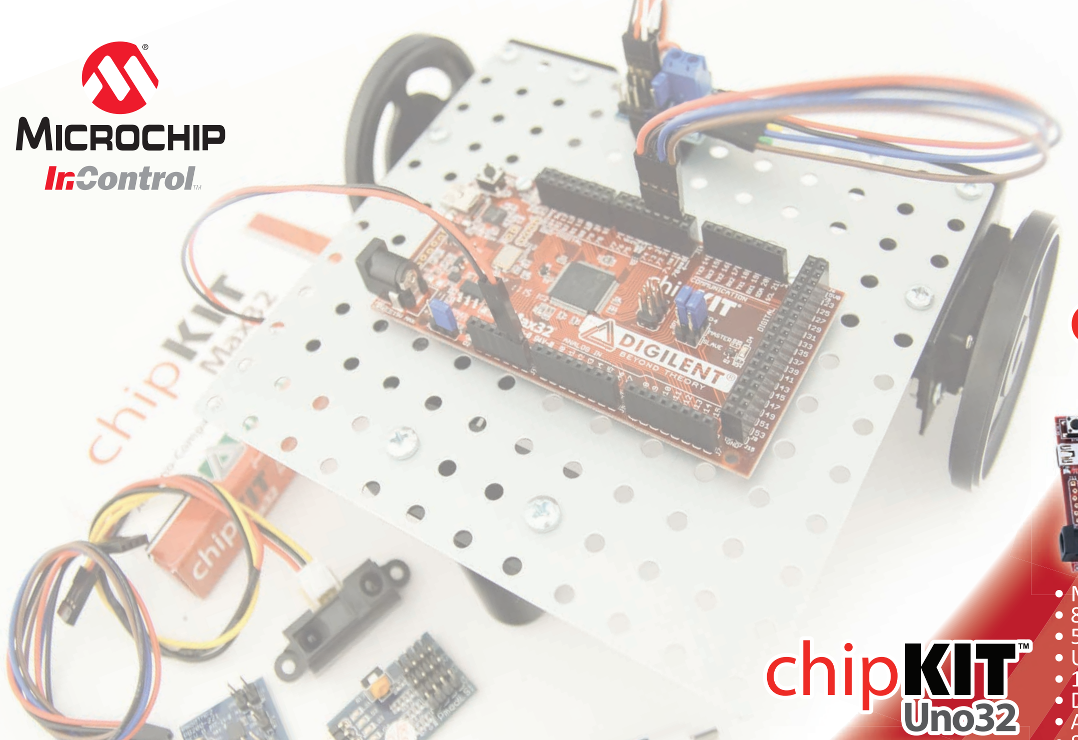
# chipKIT™

## Advanced 32-bit MCU Power for the Arduino™ Community

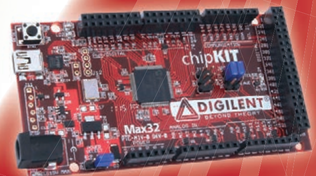
chipKIT™ development boards are the first 32-bit-microcontroller-based platforms that are compatible with many existing Arduino™ code examples, reference materials and other resources. They can be programmed using an environment based on the original Arduino™ IDE modified to support PIC32.

- Pin-out compatibility with many existing Arduino™ shields that can operate at 3.3V
- Lower price-point at four times the performance than existing solutions
- Advanced capabilities including integrated USB (Device/Host, OTG) & integrated Ethernet

MICROCHIP InControl




### chipKIT™ Max32

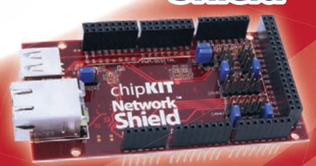



- Microchip® PIC32MX795F512
- 80 Mhz 32-bit MIPS
- 512K Flash, 128K RAM
- USB 2.0 OTG controller
- 10/100 Ethernet MAC
- Dual CAN controllers
- Arduino™ "Mega" form factor
- 83 available I/O

### chipKIT™ Uno32

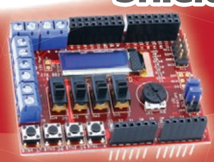

- Microchip® PIC32MX320F128
- 80 Mhz 32-bit MIPS
- 128K Flash, 16K SRAM
- Arduino™ "Uno" form factor
- 42 available I/O

### chipKIT™ Network Shield




- 10/100 Ethernet
- USB Host, Device, OTG
- Dual CAN transceivers
- Dual I²C™ connectors
- 256kbit I²C™ EEPROM

### chipKIT™ Basic I/O Shield



- 128x32 OLED Graphic Display
- Digital temperature sensor
- 256kbit EEPROM
- 4 switches, 4 push buttons, 8 LEDs
- 4 Open drain transistor outputs
- Analog potentiometer

DIGILENT® BEYOND THEORY

www.digilentinc.com/chipkit